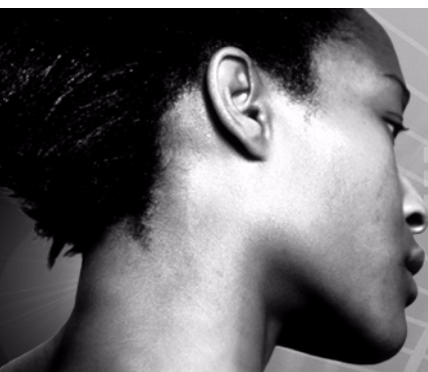




Orela™ 4500

Hardware Reference Manual

M-20401-001

Dspfactory Ltd.

March 2004

This document may not be redistributed or transmitted in any form.

Copyright © 2004 Dspfactory Ltd.

Dspfactory, SignaKlara and Orela are either trademarks or registered trademarks of Dspfactory Ltd. All other brands, product names and company names mentioned herein may be trademarks or registered trademarks of their respective holders.

It is the policy of Dspfactory Ltd. to improve products as new technology becomes available. Dspfactory Ltd., therefore, reserves the right to change specifications without prior notice.

No warranty expressed or implied.

To maintain the quality of our publications, we request your comments on the accuracy, clarity, organization, and value of this document.

Address correspondence to:

Corporate Head Office

Dspfactory Ltd.
611 Kumpf Drive, Unit 200
Waterloo, Ontario
Canada N2V 1K8
Tel: +1 519 884 9696
Fax: +1 519 884 0228

European Office

Dspfactory SA
Champs-Montants 12a
2074 Marin
Switzerland
Tel: +41 32 755 74 30
Fax: +41 32 755 74 01

Website

www.dspfactory.com

E-mail

info@dspfactory.com

TABLE OF CONTENTS

1. Introduction	1
1.1 Purpose	.1
1.2 Intended Audience	.1
1.3 Conventions	.1
1.4 Manual Organization	.1
1.5 Further Reading	.2
2. System Overview	3
2.1 System Architecture Introduction	.3
2.2 Signal Path Overview	.4
2.2.1 Input Stage	4
2.2.2 Input/Output Processor (IOP)	4
2.2.3 Weighted Overlap-Add (WOLA) Filterbank Coprocessor	4
2.2.4 Output Stage	5
2.3 Other Important System Components	.5
2.3.1 RCore	5
2.3.2 Memory	5
2.3.3 Interfaces	5
2.3.4 Peripherals	6
3. Input Stage	7
3.1 Overview and Architecture	.7
3.2 Input Selection	.8
3.2.1 Control and Configuration Registers	8
3.2.1.1 A_INPUT_CTRL Settings	8
3.3 Preamplifiers	.8
3.3.1 Control and Configuration Registers	9
3.3.1.1 A_IN_GAIN_CTRL Settings.	9
3.4 Line Out	.9
3.4.1 Control and Configuration Registers	10
3.4.1.1 A_INFILT_CTRL Settings	10

3.5 Low-Pass Filters	10
3.5.1 <i>Control and Configuration Registers</i>	10
3.5.1.1 A_INPUT_CTRL Settings	11
3.6 A/D Converters	11
3.6.1 <i>Control and Configuration Registers</i>	12
3.6.1.1 A_INPUT_CTRL Settings	12
3.6.1.2 A_ADC_CUR_CTRL Settings	12
3.6.1.3 A_DEL_INT_CTRL Settings	13
3.6.1.4 A_DEL_FRAC_CTRL Settings	14
3.7 DC Removal Filters	14
3.7.1 <i>Control and Configuration Registers</i>	14
3.7.1.1 A_INFILT_CTRL Settings	14
3.8 Decimation Filter and Input Muting	14
3.8.1 <i>Control and Configuration Registers</i>	15

4. Output Stage 17

4.1 Overview and Architecture	17
4.2 Interpolation Filters	18
4.3 D/A Converters	18
4.3.1 <i>Control and Configuration Registers</i>	19
4.3.1.1 A_OUTPUT_CTRL Settings	19
4.3.1.2 A_DAC_CTRL Settings	19
4.4 Output Reconstruction Filters	20
4.4.1 <i>Control and Configuration Registers</i>	20
4.4.1.1 A_OUTPUT_CTRL Settings	20
4.5 Output Attenuators	21
4.5.1 <i>Control and Configuration Registers</i>	21
4.5.1.1 A_OUT_ATTN_CTRL Settings	21
4.6 Direct Digital Output Drives	22
4.6.1 <i>Control and Configuration Registers</i>	25
4.6.1.1 A_OUTPUT_CTRL Settings	25
4.6.1.2 A_DAC_CTRL Settings	25
4.7 Analog Outputs	26

5. WOLA Filterbank Coprocessor

27

5.1 WOLA Filterbank Coprocessor	27
5.1.1 WOLA Control Signals	27
5.1.2 WOLA Operation	28
5.1.3 WOLA Analysis	30
5.1.3.1 Mono Analysis	30
5.1.3.2 Stereo analysis	31
5.1.4 WOLA Gain Application	32
5.1.4.1 Mono Gain Application	32
5.1.4.2 Stereo Gain Application	34
5.1.4.3 Complex Gain Application	35
5.1.4.3.1 Mono Complex Gain Application	35
5.1.4.3.2 Stereo Complex Gain Application	36
5.1.5 Synthesis	36
5.1.6 Block Floating Point	37
5.2 WOLA Energy Calculations	39
5.2.1 Mono Energy Calculations	40
5.2.2 Stereo Energy Calculations	41
5.3 WOLA Configuration	43
5.3.1 Include File	43
5.3.2 Input Block Size (R)	44
5.3.3 Window Length (L)	44
5.3.4 WOLA Performance	44
5.3.5 Block Rate (Tick)	45
5.3.6 Delay	45
5.3.7 Frequency Response	45
5.3.7.1 Filterbank Aliasing Distortion	46
5.3.8 Power Consumption	47

6. Input/Output Processor (IOP)

49

6.1 Introduction	49
6.2 IOP Mono Mode	51
6.2.1 IOP Mono Mode Smart Input FIFO	52
6.2.2 IOP Mono Mode Smart Output FIFO	53

6.2.3 IOP Mono Mode Smart FIFO Address Mapping	53
6.3 IOP Simple Stereo Mode	54
6.3.1 IOP Simple Stereo Mode Smart Input FIFO	55
6.3.2 IOP Simple Stereo Mode Smart Output FIFO	55
6.3.3 IOP Simple Stereo Mode Smart FIFO Address Mapping	55
6.4 IOP Full Stereo Mode	56
6.4.1 IOP Full Stereo Mode Smart Input FIFO	57
6.4.2 IOP Full Stereo Mode Smart Output FIFO	57
6.4.3 IOP Full Stereo Mode Smart FIFO Address Mapping	57
6.5 IOP Digital Mixed Mode	58
6.5.1 IOP Digital Mixed Mode Smart Input FIFO	59
6.5.2 IOP Digital Mixed Mode Smart Output FIFO	59
6.5.3 IOP Digital Mixed Mode Smart FIFO Address Mapping	59
6.6 IOP Auto-Muting	60
6.7 IOP Enable/Disable and Reset.	61
6.8 Control and Configuration Registers	61
6.8.1 D_IO_PROC_CFG Settings	61
6.8.2 D_SYS_CTRL Settings	62

7. Memory 63

7.1 Architecture	63
7.2 Program Memory	64
7.3 Data Memory	65
7.4 Memory Maps	65
7.5 Data ROM	66
7.6 Program ROM	66
7.7 Memory Control and Configuration Registers	67

8. Interfaces 69

8.1 General Purpose I/O (GPIO)	69
8.1.1 Interrupt	69
8.1.2 Control and Configuration Registers	70
8.1.2.1 D_GPIO_PIN_CFG Settings	71

8.1.2.2 D_GPIO_INT_CFG Settings	73
8.2 UART	73
8.2.1 Interrupts	74
8.2.2 Control and Configuration Registers	74
8.2.2.1 D_UART_CFG Settings	74
8.3 I2S Interface	75
8.4 Low-Speed A/D Converters	77
8.4.1 Control and Configuration Registers	79
8.4.1.1 A_LSAD_<channel>_CTRL Settings	79
8.4.1.2 A_DIV_CLK_CTRL Settings	80
8.5 Pulse Code Modulation (PCM) Interface	80
8.5.1 Interrupts	82
8.5.2 Operation	82
8.5.3 Data Transfer Formatting Options	83
8.5.4 Control and Configuration Registers	84
8.5.4.1 D_PCM_CTRL Settings	85
8.5.4.2 D_PCM_CFG Settings	85
8.6 SPI Port	86
8.6.1 Interrupts	87
8.6.2 Control and Configuration Registers	87
8.6.2.1 A_PSU_CTRL Settings	87
8.6.2.2 D_SPI_CTRL Settings	88
8.6.2.3 D_SPI_CFG Settings	89
8.7 TWSS Interface	89
8.7.1 Operation	89
8.7.1.1 Master to Slave Transfer Initiation	89
8.7.1.2 TWSS Receive Mode	90
8.7.1.3 TWSS Transmit Mode	90
8.7.2 Interrupts	91
8.7.3 Control and Configuration Registers	91
8.7.3.1 D_TWSS_CTRL Settings	92
8.7.3.2 D_TWSS_CFG Settings	92
8.8 Debug Port UART	93
8.8.1 Baud Rate Synchronization	93
8.8.2 Interrupts	94
8.8.3 Audio Streaming	94

8.8.4	<i>Control and Configuration Registers</i>	94
8.8.4.1	<i>D_DBG_AUDIO_MODE_CTRL Settings</i>	95
8.8.4.2	<i>D_ACCESS_CFG Settings</i>	95
8.9	SSP_CFG Interface	95
8.9.1	<i>Control and Configuration Registers</i>	97
8.9.1.1	<i>D_SSP_CFG_CTRL Settings</i>	97
8.10	Wireless Receiver Remote Control Interface	98
8.10.1	<i>Interrupts</i>	99
8.10.2	<i>Control and Configuration Registers</i>	99
8.10.2.1	<i>D_REMOTE_DATA Settings</i>	99
8.10.2.2	<i>D_REMOTE_CFG Settings</i>	100

9. Peripherals 101

9.1	Interrupt Controller	101
9.1.1	<i>Interrupt Handling</i>	101
9.1.2	<i>Interrupt Configuration</i>	101
9.1.3	<i>Interrupt Priority</i>	102
9.1.4	<i>Automatic Priority Rotation</i>	103
9.1.5	<i>Automatic Acknowledge</i>	103
9.1.6	<i>Control and Configuration Registers</i>	103
9.1.6.1	<i>D_INT_CTRL Settings</i>	103
9.1.6.2	<i>D_INT_STATUS Settings</i>	104
9.1.6.3	<i>D_INT_EBL Settings</i>	105
9.2	Battery Monitor	106
9.2.1	<i>Control and Configuration Registers</i>	107
9.3	Oscillator Circuitry and Clock Generation	107
9.3.1	<i>On-chip Oscillator Configuration and Calibration</i>	109
9.3.2	<i>Clocks and Clock Domains</i>	110
9.3.3	<i>Sampling Frequency Configuration</i>	111
9.3.4	<i>Control and Configuration Registers</i>	112
9.3.4.1	<i>A_CCR_DATA Settings</i>	112
9.3.4.2	<i>A_CLK_CTRL Settings</i>	112
9.3.4.3	<i>A_ADC_CTRL Settings</i>	113
9.3.4.4	<i>D_CLK_SEL_CFG Settings</i>	114
9.3.4.5	<i>D_CLKDIV_CFG Settings</i>	115

9.3.4.6	D_USRCLKDIV_CFG Settings	116
9.3.4.7	D_WOLADIV_CFG Settings	116
9.3.4.8	D_STANDBY_CFG Settings	117
9.4	General-Purpose Timer	117
9.4.1	Configuration and Operation	118
9.4.2	Starting the Timer	118
9.4.3	Stopping the Timer	118
9.4.4	Control and Configuration Registers	118
9.4.4.1	D_SYS_CTRL Settings	118
9.4.4.2	D_TIMER_CFG Settings	118
9.5	Multi-Chip Synchronization	119
9.6	Power-On Reset	120
9.7	Power Supply Regulation and Management	121
9.7.1	Soft Power Down Mode	122
9.7.2	Power Management Unit	123
9.7.3	Control and Configuration Registers	124
9.7.3.1	A_DIV_CLK_CTRL Settings	124
9.7.3.2	A_PSU_CTRL Settings	124
9.7.3.3	D_STANDBY_CFG Settings	125
9.8	Watchdog Timer	125
9.8.1	Operation	125
9.8.2	Control and Configuration Registers	126
9.8.2.1	D_SYS_CTRL Settings	126
9.8.2.2	D_WATCHDOG_CFG Settings	127
9.9	Configuration Register Controller	127
9.9.1	Control and Configuration Registers	127
9.9.1.1	D_CFG_REG_CTRL Settings	128

A. Sampling Frequencies 129

A.1	Sampling Frequency Overview	129
------------	--	------------

B. PCM Timing Diagrams 131

B.1	Timing Diagrams Explanation	131
B.2	16-Bit Timing Diagrams	131

B.3 32-Bit Timing Diagrams	135
<i>C. Control and Configuration Register Overview</i>	<i>141</i>

Introduction

1.1 PURPOSE

This manual provides a hardware reference for application developers working with Orela™ 4500, which is based on SignaKlara™ 2.5 technology, referred to as SK2.5. The manual describes all functional features available on the Orela 4500 chip, how they are used and how they are configured.

The Orela 4500 Hardware Reference Manual is part of the Orela Evaluation and Development Kit (Orela EDK) 2.5.

1.2 INTENDED AUDIENCE

This manual is intended for engineers and anyone else responsible for creating and maintaining audio processing applications based on Orela 4500.

This manual assumes that you are familiar with programming, digital signal processing (DSP) application concepts and DSP tools.

1.3 CONVENTIONS

This book uses the following conventions:

- Control and configuration registers are shown in a special font. All control register names are prefixed with the letter “A” or the letter “D”. The letter “A” indicates that the control register is associated with control and configuration of an analog entity on the chip whereas the letter “D” indicates that the control register is associated with control and configuration of a digital entity on the chip.
- Angle brackets “<” and “>” identify an optional parameter or indicate a placeholder for specific information. To use an optional parameter or replace a placeholder, specify the information within the brackets; do not include the brackets themselves.
- Default settings for registers and bit fields are marked with an asterisk (*).
- All sample rates specified are the final decimated sample rates, unless stated otherwise.
- In general, numbers are presented in decimal notation. In cases where hexadecimal or binary notation is more convenient, these numbers are identified by the prefixes “0x” and “0b” respectively. For example, the decimal number 123456 can also be represented as 0x1E240 or 0b11110001001000000.

1.4 MANUAL ORGANIZATION

This manual contains the following chapters:

1. **Chapter 1: Introduction**, describes the purpose of the manual, outlines how the reference manual is organized and gives a list of suggested readings for more information.

2. **Chapter 2: System Overview**, provides an architectural overview of Orela 4500 by describing the audio processing chain.
3. **Chapter 3: Input Stage**, contains a detailed description of the input stage and how to configure it using the associated control and configuration registers.
4. **Chapter 4: Output Stage**, contains a detailed description of the output stage and how to configure it using the associated control and configuration registers.
5. **Chapter 5: WOLA Filterbank Coprocessor**, describes in detail how to configure and use the Weighted Overlap-Add (WOLA) Filterbank coprocessor.
6. **Chapter 6: Input/Output Processor (IOP)**, describes the Input/Output Processor (IOP), how to access incoming and outgoing samples, and how to configure the various audio processing modes.
7. **Chapter 7: Memory**, details the memory architecture and all hardware memory maps.
8. **Chapter 8: Interfaces**, describes all interfaces, how to connect to them and how to configure them for the desired operation.
9. **Chapter 9: Peripherals**, describes all peripherals, including the power supply and the on-chip oscillator, and explains how to select a desired sampling frequency.
10. **Appendix A: Sampling Frequencies**, presents an overview of possible sampling frequencies and the system settings required to obtain each of these frequencies.
11. **Appendix B: PCM Timing Diagrams**, provides example functional timing diagrams for certain configurations of the PCM interface.
12. **Appendix C: Control and Configuration Register Overview**, presents an overview of all control and configuration registers.

1.5 FURTHER READING

For more information, refer to the following documents:

- *Orela EDK 2.5 Getting Started Guide*
- *Orela 4500 Evaluation and Development Board Manual*
- *Orela 4500 Firmware Reference Manual*
- *Orela EDK 2.5 Software Tools Manual*
- *LLCOM Audio Extensions for NOAHlink Hardware and Software Reference*
- *RCORE DSP Architecture Manual*
- *SignaKlara Debugger User's Guide* (available as an online Help file)
- *SignaKlara 2.5 Communication Protocols Manual*
- Orela 4500 series datasheets

System Overview

2.1 SYSTEM ARCHITECTURE INTRODUCTION

Orela 4500 contains many analog and digital processing mechanisms, which work together to provide a complete audio processing chain. The analog blocks include an input stage with two 16-bit analog-to-digital (A/D) converters. The two on-chip direct digital outputs can be configured separately, or they can be used in combination to create a high-power output. Both output drivers enable the use of passive speakers (receivers), eliminating the need for external amplifiers.

Among other peripherals, Orela 4500 features internal clock generation and power regulation for excellent noise and power performance. Two DSP subsystems operate concurrently: the RCore, a fully programmable DSP core, and the Weighted Overlap-Add (WOLA) filterbank coprocessor, a dedicated, configurable processor that transforms signals to time-frequency domain and/or performs other vector-based computations.

Orela 4500 has multiple interfaces, both analog and digital, that enable a wide variety of transducers, switches, potentiometers, and other systems to be connected to Orela. In addition to the RCore and the WOLA, several other units optimize the architecture to audio processing, such as the Input/Output Processor (IOP)—an audio-targeted DMA processor that runs in the background and manages the data flow between the converters, the RCore and the WOLA.

A simplified illustration of Orela 4500 is shown in Figure 1.

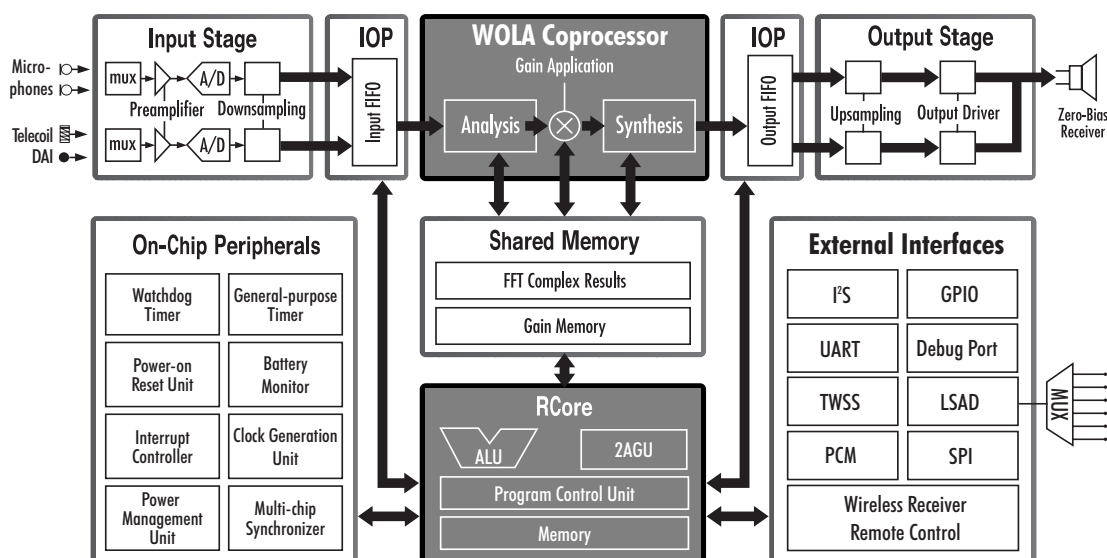


Figure 1: Simplified system overview of Orela 4500

2.2 SIGNAL PATH OVERVIEW

This section describes the processing of sound signals that enter the Orela 4500 chip through one or more of the input means shown in Figure 1.

2.2.1 Input Stage

The incoming sound signals enter one or more audio transducers (microphones or telecoils) that convert the incoming signals to electrical signals, which are passed into the audio inputs of the Orela 4500 chip. Although up to four audio inputs can be connected to the chip at the same time, only two channels can be processed simultaneously. Two multiplexers route two of the four incoming signals to each of the two processing channels in the input stage. After multiplexing, the desired signals are amplified with a specified gain value in the two preamplifiers. Subsequently, the signals go through two second-order anti-aliasing filters (one filter for each channel) where undesired high-frequency components are removed to prepare the signal for A/D conversion. The A/D conversion happens in the two oversampling A/D converters, which both have a configurable sampling frequency. Following A/D conversion, the signals are decimated to the final desired sampling frequency, and residual DC-content in the input signals is removed.

2.2.2 Input/Output Processor (IOP)

After passing through the input stage, the digital signal is stored in an input FIFO from where it can be accessed by either of the two other on-chip processors (RCore, WOLA). The task of storing data in the input FIFO is automatically managed by the IOP. Similarly, the task of retrieving data from an output FIFO and delivering it to the output stage is also managed by the IOP.

The IOP has several operating modes to accommodate a variety of applications. Depending on the configuration, the IOP can handle a signal in one of the following ways:

mono	One input channel is processed and one output signal is produced
simple stereo	Two input channels are processed and one output signal is produced
full stereo	Two input channels are processed and two output signals are produced
digital mixed	Two input channels are processed and one output signal is produced

2.2.3 Weighted Overlap-Add (WOLA) Filterbank Coprocessor

The WOLA Filterbank Coprocessor performs processing on data stored in the input FIFO and stores the processed data in the output FIFO. The signal processing in the WOLA typically starts with an analysis task where conversion of the input signal(s) to a time-frequency domain representation is performed, followed by gain application where appropriate gains are added to shape the input signals in a desirable fashion, and ending with a synthesis task where the processed signal is converted back to a time-domain representation suitable for processing in the output stage. Numerous configuration options for the WOLA exist that allow trade-offs of processing power requirements vs. audio fidelity performance.

2.2.4 Output Stage

When the processed data is ready in the output FIFO, it is sent to one (or both) of the two output channels. In the output stage, the output signal is converted to an analog representation or directed to the output drivers.

If the analog output is to be used, the signal passes through a reconstruction filter with a programmable cut-off frequency. As well, the output can be attenuated or muted.

Otherwise the signal is fed to the output drivers.

2.3 OTHER IMPORTANT SYSTEM COMPONENTS

Other components are essential for controlling and enhancing the signal processing. These are described in the sections below.

2.3.1 RCore

The RCore is a dual-Harvard architecture, 16-bit programmable fixed-point DSP core that provides the master control functionality for the entire system. Its characteristics allows it to perform powerful operations that control how the signal is processed (or processes the signal directly). The RCore is documented separately in the *RCore DSP Architecture Manual*.

2.3.2 Memory

The memories on Orela 4500 are as follows:

- 12-Kword program memory (PRAM)
- Two 4-Kword data memories (XRAM and YRAM)
- Two 384-word dual-port FIFO memories (FIFO RAM)
- Two 128-word dual-port 18-bit memories dedicated to WOLA output results (TEMP RAM)
- 576-word memory dedicated to WOLA gain values, WOLA windows, and other configuration data (CONF RAM)

2.3.3 Interfaces

There are various ways to interface to Orela 4500, which include:

- A general purpose input/output (GPIO) interface
- Universal asynchronous receiver/transmitter (UART)
- An I²S interface
- Low-speed A/D (LSAD) inputs
- A pulse-code modulation (PCM) interface
- A serial peripheral interface (SPI)
- Two-wire synchronous serial (TWSS) interface
- A debug port
- A wireless remote control interface

2.3.4 Peripherals

Peripherals on Orela 4500 include:

- An interrupt controller
- A battery monitor
- Oscillator circuitry and clock generation unit
- A general-purpose timer
- Multi-chip synchronizer
- Power-on reset unit
- Power supply regulation and management unit
- A watchdog timer

3.1 OVERVIEW AND ARCHITECTURE

The input stage provides signal selection, signal conditioning (amplification and anti-alias filtering), analog-to-digital conversion, and sample-rate decimation.

Specifically, the input stage provides:

- Input selection via analog multiplexers allowing full flexibility with input source to channel assignments
- Programmable input amplification to adjust input signals to suitable levels for the A/D converters
- Anti-aliasing filters with a 20 kHz -3 dB cut-off frequency. The filters are activated at all gain settings (including preamp bypass) unless they are specifically disabled by configuring the control register appropriately
- Over-sampled analog-to-digital converters with programmable sample rates
- High-quality decimation filtering with optional gain adjustments at all selectable sampling rates
- Selectable input muting on both channels

The following figure illustrates the input stage:

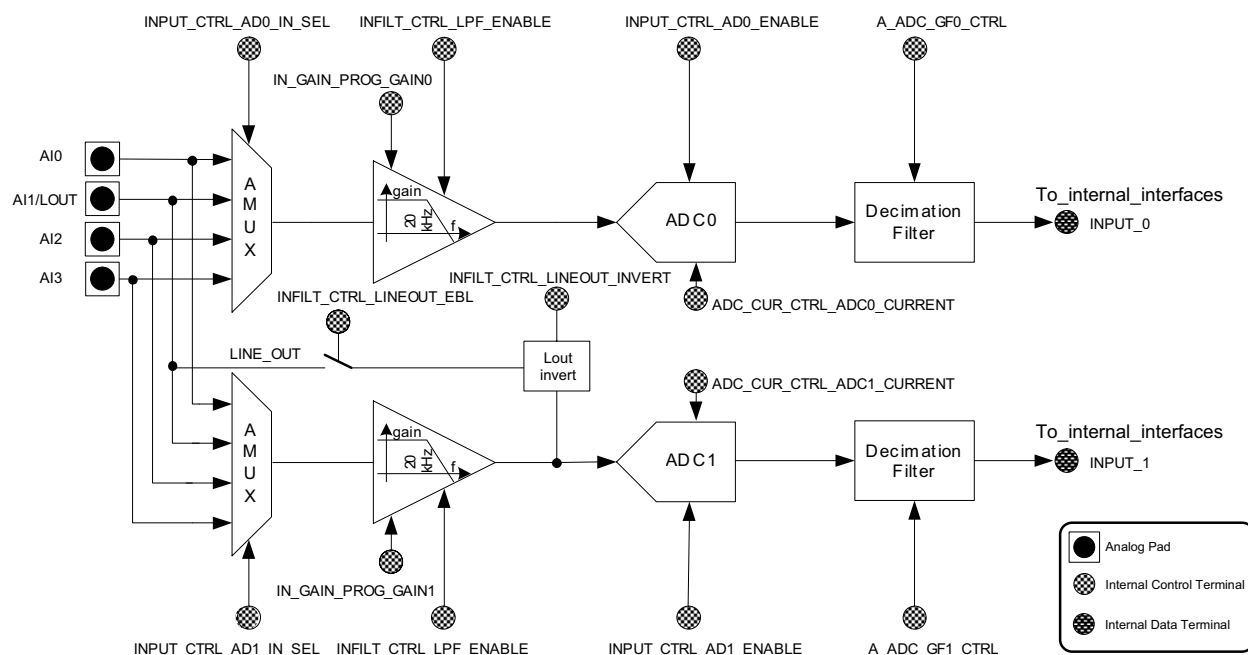


Figure 2: Input stage overview

3.2 INPUT SELECTION

Signal selection for each of the two input channels is done in the A_INPUT_CTRL control register using the INPUT_CTRL_AD0_IN_SEL and INPUT_CTRL_AD1_IN_SEL bit fields. Four possible input sources are available for each channel. Both channels should not be configured to use the same input source or a reduction in the signal quality for both channels will occur due to an associated reduction of input impedance. External de-coupling capacitors in the range of 5 to 10 nF are required for all inputs. Unused inputs should be grounded.

3.2.1 Control and Configuration Registers

Register Name	Register Description	Address
A_INPUT_CTRL	Select inputs for the two channels	A:0x16

3.2.1.1 A_INPUT_CTRL Settings

Bit Field	Field Name	Field Description
6:5	INPUT_CTRL_AD1_IN_SEL	Select input for ADC1
2:1	INPUT_CTRL_AD0_IN_SEL	Select input for ADC0

Field Name	Value Symbol	Value Description	Hex Value
INPUT_CTRL_AD1_IN_SEL	AD1_SEL_AI0	Select input AI0	0x0
	AD1_SEL_AI1	Select input AI1	0x1
	AD1_SEL_AI2*	Select input AI2	0x2
	AD1_SEL_AI3	Select input AI3	0x3
INPUT_CTRL_AD0_IN_SEL	AD0_SEL_AI0*	Select input AI0	0x0
	AD0_SEL_AI1	Select input AI1	0x1
	AD0_SEL_AI2	Select input AI2	0x2
	AD0_SEL_AI3	Select input AI3	0x3

Note: One input cannot be applied to both channels simultaneously.

3.3 PREAMPLIFIERS

The purpose of the two preamplifiers is to amplify the input signals. When enabled, these preamplifiers can amplify the signal between 12 and 30 dB in 3 dB steps. To select a gain value for each of the two preamplifiers, use the A_IN_GAIN_CTRL control register, using the IN_GAIN_PROG_GAIN0 and IN_GAIN_PROG_GAIN1 bit fields.

3.3.1 Control and Configuration Registers

Register Name	Register Description	Address
A_IN_GAIN_CTRL	Select gain values for the two preamplifiers	A:0x17

3.3.1.1 A_IN_GAIN_CTRL Settings

Bit Field	Field Name	Field Description
6:4	IN_GAIN_PROG_GAIN1	Select gain value for preamplifier 1
2:0	IN_GAIN_PROG_GAIN0	Select gain value for preamplifier 0

Field Name	Value Symbol	Value Description	Hex Value
IN_GAIN_PROG_GAIN1	PGAIN1_BYPASS	Bypass preamplifier	0x0
	PGAIN1_12DB	Set gain to 12 dB	0x1
	PGAIN1_15DB	Set gain to 15 dB	0x2
	PGAIN1_18DB*	Set gain to 18 dB	0x3
	PGAIN1_21DB	Set gain to 21 dB	0x4
	PGAIN1_24DB	Set gain to 24 dB	0x5
	PGAIN1_27DB	Set gain to 27 dB	0x6
	PGAIN1_30DB	Set gain to 30 dB	0x7
IN_GAIN_PROG_GAIN0	PGAIN0_BYPASS	Bypass preamplifier	0x0
	PGAIN0_12DB	Set gain to 12 dB	0x1
	PGAIN0_15DB	Set gain to 15 dB	0x2
	PGAIN0_18DB*	Set gain to 18 dB	0x3
	PGAIN0_21DB	Set gain to 21 dB	0x4
	PGAIN0_24DB	Set gain to 24 dB	0x5
	PGAIN0_27DB	Set gain to 27 dB	0x6
	PGAIN0_30DB	Set gain to 30 dB	0x7

3.4 LINE OUT

The amplified and low-pass-filtered input signal on channel one is available on the AI1/LOUT pad if the pad is configured for this in the A_INFILT_CTRL control register using the INFILT_CTRL_LINEOUT_EBL bit field; otherwise, the pad is used as a selectable input for either channel. If desired, it is also possible to invert the output from the line-out signal by setting the INFILT_CTRL_LINEOUT_INV bit in A_INFILT_CTRL. Enabling this bit results in a 180° phase shift of the input signal.

3.4.1 Control and Configuration Registers

Register Name	Register Description	Address
A_INFILT_CTRL	Enable or disable and configure line-out signal	A:0x12

3.4.1.1 A_INFILT_CTRL Settings

Bit Field	Field Name	Field Description
1	INFILT_CTRL_LINEOUT_EBL	Enable/disable line-out signal
0	INFILT_CTRL_LINEOUT_INV	Select regular or inverted line-out signal

Field Name	Value Symbol	Value Description	Hex Value
INFILT_CTRL_LINEOUT_EBL	LINE_OUT_DISABLE*	Disable line-out signal	0x0
	LINE_OUT_ENABLE	Enable line-out signal	0x1
INFILT_CTRL_LINEOUT_INV	LINE_OUT_INVERT_DISABLE*	Regular line-out signal	0x0
	LINE_OUT_INVERT_ENABLE	Inverted line-out signal	0x1

3.5 LOW-PASS FILTERS

The input anti-aliasing filters (combined with the preamplifiers as shown in Figure 2 on page 7) when enabled can attenuate high-frequency aliasing components of the input signals, which are the side-effect of digital sampling. Both filters are based on a 2nd-order, active low-pass filter with a -3 dB cut-off frequency at 20 kHz and may be enabled using the INFILT_CTRL_LPF_ENABLE bit in the A_INFILT_CTRL control register. It should be noted that the two anti-aliasing filters cannot be configured independently.

The A/D converters oversample the input signal by a factor of eight. As a result, if a 16 kHz sampling frequency is selected with MCLK at 1.28 MHz, no aliased images will appear for input frequencies below 120 kHz. Thus, the roll-off of the low-pass filters at the cut-off frequency of 20 kHz combined with the roll-off of typical microphones will provide sufficient aliasing protection.

3.5.1 Control and Configuration Registers

Register Name	Register Description	Address
A_INFILT_CTRL	Enable/disable the input anti-aliasing low-pass filters	A:0x12

3.5.1.1 A_INPUT_CTRL Settings

Bit Field	Field Name	Field Description
2	INFILT_CTRL_LPF_ENABLE	Enable/disable low-pass filtering

Field Name	Value Symbol	Value Description	Hex Value
INFILT_CTRL_LPF_ENABLE	LPF_IN_BYPASS	Bypass input LPF	0x0
	LPF_IN_ACTIVE_20KHZ*	Enable active low-pass filter	0x1

3.6 A/D CONVERTERS

Each A/D converter oversamples the input signals by a factor of eight and can be enabled or disabled in the A_INPUT_CTRL control register using the INPUT_CTRL_AD0_ENABLE and INPUT_CTRL_AD1_ENABLE bit fields. Note that the high oversampling will shift aliasing components higher in the frequency band (e.g. if a 16 kHz sampling frequency is selected with MCLK at 1.28 MHz, no aliased images will appear for input frequencies below 120 kHz). If A/D converter zero (ADC0) is disabled and A/D converter one (ADC1) is enabled, the data read through ADC1 will be treated as though it is from ADC0 by the Input/Output Processor and the rest of the digital system (see Chapter 6, “Input/Output Processor (IOP)” on page 49).

The selection of a desired sampling frequency is tied closely to the configuration of the oscillator circuitry and the clock distribution on the chip. For a detailed description of how to select a desired sampling frequency, see Section 9.3, “Oscillator Circuitry and Clock Generation” on page 107.

The current supplied to the A/D converters is configured in the A_ADC_CUR_CTRL control register using the ADC_CUR_CTRL_ADC0_CURRENT and ADC_CUR_CTRL_ADC1_CURRENT bit fields. In general, when sampling frequencies below 20 kHz are desired, the nominal value (see below) for these bit fields should be used. For higher sampling frequencies, a higher current should be applied to prevent any reduction in fidelity. The nominal current I_{nom} is approximately 40 μ A at 1.28 MHz operation, with a supply voltage of 1.25 V.

The ADC_CUR_CTRL_HIGH_FREQ_EBL bit in the A_ADC_CUR_CTRL control register should be set when using sampling frequencies associated with MCLK frequencies above 1.28 MHz. For more information about setting the MCLK frequency, see Section 9.3, “Oscillator Circuitry and Clock Generation” on page 107.

If both A/D converters are enabled, the two control registers A_DEL_INT_CTRL and A_DEL_FRAC_CTRL can be used to set a fixed delay between samples in the two converters. Applying non-zero values in the two register fields DEL_FRAC_CTRL_DEL_FRAC_POS in the A_DEL_FRAC_CTRL register and DEL_INT_CTRL_DEL_INT_POS in the A_DEL_INT_CTRL register enables this feature. The field DEL_INT_CTRL_LEADER selects whether the sampling of values in ADC0 is delayed relative to the sampling of values in ADC1 or vice versa. The total delay D_{tot} of samples performed in channel one relative to samples performed in channel zero (expressed in seconds) can be determined with the following equation:

$$D_{tot} = \frac{(DEL_FRAC_CTRL_DEL_FRAC + DEL_INT_CTRL_DEL_INT \cdot (ADC_CTRL_SAMPLE_FREQ + 8))}{f_{MCLK}}$$

3.6.1 Control and Configuration Registers

Register Name	Register Description	Address
A_INPUT_CTRL	Input channel configuration	A:0x16
A_ADC_CUR_CTRL	Select current to A/D converters	A:0x1C
A_DEL_INT_CTRL	Sample clock integer delay configuration and control	A:0x21
A_DEL_FRAC_CTRL	Sample clock delay fractional control	A:0x22

3.6.1.1 A_INPUT_CTRL Settings

Bit Field	Field Name	Field Description
4	INPUT_CTRL_AD1_ENABLE	Enable/disable A/D converter 1
0	INPUT_CTRL_AD0_ENABLE	Enable/disable A/D converter 0

Field Name	Value Symbol	Value Description	Hex Value
INPUT_CTRL_AD1_ENABLE	AD1_DISABLE*	Disable A/D converter 1	0x0
	AD1_ENABLE	Enable A/D converter 1	0x1
INPUT_CTRL_AD0_ENABLE	AD0_DISABLE	Disable A/D converter 0	0x0
	AD0_ENABLE*	Enable A/D converter 0	0x1

3.6.1.2 A_ADC_CUR_CTRL Settings

Bit Field	Field Name	Field Description
7	ADC_CUR_CTRL_HIGH_FREQ_EBL	Enable/disable high MCLK frequency current control.
5:3	ADC_CUR_CTRL_ADC1_CURRENT	Select current for A/D converter 1
2:0	ADC_CUR_CTRL_ADC0_CURRENT	Select current for A/D converter 0

Field Name	Value Symbol	Value Description	Hex Value
ADC_CUR_CTRL_HIGH_FREQ_EBL	ADC_HIGH_FREQUENCY_DISABLE	Disable high MCLK frequency current control	0x0
	ADC_HIGH_FREQUENCY_ENABLE	Enable high MCLK frequency current control	0x1
ADC_CUR_CTRL_ADC1_CURRENT	ADC1_CURRENT_4_0X	Select 4 x Inom	0x0
	ADC1_CURRENT_2_0X	Select 2 x Inom	0x1
	ADC1_CURRENT_1_3X	Select 1.3 x Inom	0x2
	ADC1_CURRENT_1_0X*	Select Inom	0x3
	ADC1_CURRENT_0_8X	Select 0.8 x Inom	0x4
	ADC1_CURRENT_0_7X	Select 0.7 x Inom	0x5
	ADC1_CURRENT_0_6X	Select 0.6 x Inom	0x6
	ADC1_CURRENT_0_5X	Select 0.5 x Inom	0x7
ADC_CUR_CTRL_ADC0_CURRENT	ADC0_CURRENT_4_0X	Select 4 x Inom	0x0
	ADC0_CURRENT_2_0X	Select 2 x Inom	0x1
	ADC0_CURRENT_1_3X	Select 1.3 x Inom	0x2
	ADC0_CURRENT_1_0X*	Select Inom	0x3
	ADC0_CURRENT_0_8X	Select 0.8 x Inom	0x4
	ADC0_CURRENT_0_7X	Select 0.7 x Inom	0x5
	ADC0_CURRENT_0_6X	Select 0.6 x Inom	0x6
	ADC0_CURRENT_0_5X	Select 0.5 x Inom	0x7

3.6.1.3 A_DEL_INT_CTRL Settings

Bit Field	Field Name	Field Description
7	DEL_INT_CTRL_LEADER	Select AD0 or AD1 as the leading sample
2:0	DEL_INT_CTRL_DEL_INT	Sample clock integer delay

Field Name	Value Symbol	Value Description	Hex Value
DEL_INT_CTRL_LEADER	DEL_INT_CTRL_AD0_LEAD*	Enable AD0 as the leading sample	0x0
	DEL_INT_CTRL_AD1_LEAD	Enable AD1 as the leading sample	0x1

3.6.1.4 A_DEL_FRAC_CTRL Settings

Bit Field	Field Name	Field Description
5:0	DEL_FRAC_CTRL_DEL_FRAC	Sample clock fractional delay

3.7 DC REMOVAL FILTERS

A DC removal filter, used to remove any DC signal component from the input signal, is part of the decimation filter for each channel; see Section 3.8, “Decimation Filter and Input Muting” on page 14.

The DC removal filters are based on 3rd-order high-pass wave digital filter structures with programmable cut-off frequencies. The cut-off frequency is selected in the A_INFILT_CTRL control register using the INFILT_CTRL_DC_REMOVE bit field. The cut-off frequency is the same for each input channel.

3.7.1 Control and Configuration Registers

Register Name	Register Description	Address
A_INFILT_CTRL	Select DC removal cut-off frequency	A:0x12

3.7.1.1 A_INFILT_CTRL Settings

Bit Field	Field Name	Field Description
5:4	INFILT_CTRL_DC_REMOVE	Select cut-off frequency

Field Name	Value Symbol	Value Description	Hex Value
INFILT_CTRL_DC_REMOVE	DC_REMOVE_CUTOFF_5HZ	5 Hz cut-off frequency	0x0
	DC_REMOVE_CUTOFF_10HZ	10 Hz cut-off frequency	0x1
	DC_REMOVE_CUTOFF_20HZ*	20 Hz cut-off frequency	0x2
	DC_REMOVE_BYPASS	Bypass filter	0x3

3.8 DECIMATION FILTER AND INPUT MUTING

The decimation filters downsample and filter the oversampled signal from the A/D converters to obtain a final non-oversampled stream of input data. Both filters are based on 11th-order low-pass wave digital filter structures.

The gain factor for each decimation filter is configured by setting the desired 8-bit, two's-complement gain values in the A_ADC_GF0_CTRL and A_ADC_GF1_CTRL control registers. The gain factor allows the input signal to be adjusted in < 0.5 dB steps. In most cases, it is desirable to

set the gain value for each filter in such a way that the signal level, after downsampling and filtering, is the same as before downsampling and filtering (i.e. no gain or attenuation performed). These gain factors are useful to fine tune gains in the input stage, and are commonly used to match gains between microphones (in a multiple-microphone system) and to compensate for changes in sampling frequency.

The following formulas give the values for the gain factors that result in an unchanged signal level after downsampling (truncating of the result may be necessary):

$$A_ADC_GF0_CTRL = 1024 / (ADC_CTRL_SAMPLE_FREQ + 8)$$

$$A_ADC_GF1_CTRL = 1024 / (ADC_CTRL_SAMPLE_FREQ + 8)$$

For instance, if MCLK equals 1.28 MHz and the desired sampling frequency is 16 kHz (ADC_CTRL_SAMPLE_FREQ bit field in the A_ADC_CTRL control register set to 0x2) the gain factor that results in equal signal level before and after decimation is 0x66.

Note: A description of MCLK and how the A_ADC_CTRL control register relates to sampling frequency selection are explained in more detail in Section 9.3, "Oscillator Circuitry and Clock Generation" on page 107.

The following formulas give the decimation filter gains (Gain_Ch0 for channel zero and Gain_Ch1 for channel one) as a function of the 8-bit, two's-complement gain values specified in the A_ADC_GF0_CTRL and A_ADC_GF1_CTRL control registers:

$$Gain_Ch0 = (ADC_CTRL_SAMPLE_FREQ + 8) \cdot A_ADC_GF0_CTRL / 1024$$

$$Gain_Ch1 = (ADC_CTRL_SAMPLE_FREQ + 8) \cdot A_ADC_GF1_CTRL / 1024$$

Note: Due to the fact that the gain values are specified using two's-complement notation, phase inversion of the input signal will take place for gain values in the interval 0x80 to 0xFF (the input signal gets multiplied by a negative value).

Setting A_ADC_GF0_CTRL to zero results in input channel zero being muted (zeros are transmitted from the input stage to the Input/Output Processor). Similarly, setting A_ADC_GF1_CTRL to zero mutes input channel one. The default value for each of A_ADC_GF0_CTRL and A_ADC_GF1_CTRL is 0x65 which results in just slightly less than unity gain.

3.8.1 Control and Configuration Registers

Register	Description	Address
A_ADC_GF0_CTRL	Decimation filter gain value for channel 0	A:0x1F
A_ADC_GF1_CTRL	Decimation filter gain value for channel 1	A:0x20

This page left blank for printing purposes.

4.1 OVERVIEW AND ARCHITECTURE

The output stage provides upsampling, interpolation filtering, digital-to-analog conversion, reconstruction filtering, output attenuation (to match Orela 4500 output levels to inputs on other external devices) and output muting. As well, the output stage provides two pulse-density modulation (PDM) based direct digital output drives that are capable of driving passive loudspeakers and receivers, and may be operated either individually or in a combined fashion to obtain higher output sound pressure levels.

Specifically, the output stage provides:

- High-quality interpolation filtering that automatically adjusts to the selected sample rate (no manual configuration required)
- High-fidelity, and oversampled D/A converters with programmable sample rate (sample rate is the same as that of the A/D converters)
- Programmable output attenuators to set output signals to suitable levels for connection with other devices
- Reconstruction filters with programmable cut-off frequencies
- Selectable output muting (analog) on both channels (part of the attenuator)
- Two direct digital outputs for zero-bias hearing aid receivers
- Selectable high-power mode that combines the two direct digital outputs for a maximum power output signal

Figure 3 illustrates the output stage.

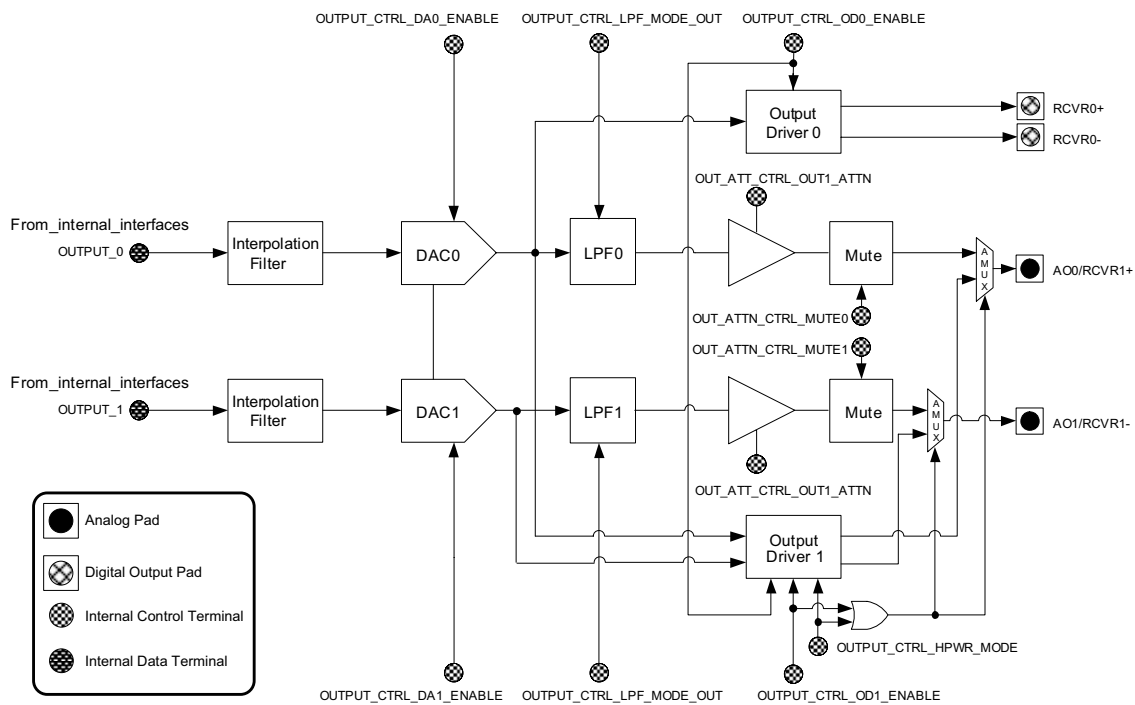


Figure 3: Output stage overview.

4.2 INTERPOLATION FILTERS

The interpolation filter on each channel serves the purpose of upsampling and filtering the signal that is passed to the D/A converters. The filters are based on 11th-order low-pass wave digital filter structures.

The interpolation filters track the upsampled frequency automatically. No user configurations are associated with this block.

4.3 D/A CONVERTERS

The D/A converters convert the digital output signals (in PDM format) back to an analog representation. Each D/A converter can be enabled or disabled in the A_OUTPUT_CTRL control register using the OUTPUT_CTRL_DA0_ENABLE and OUTPUT_CTRL_DA1_ENABLE bit fields. The D/A converters always work at the same sampling frequency as the A/D converters. For a detailed description of how to select a desired sampling frequency, see Section 9.3, "Oscillator Circuitry and Clock Generation" on page 107.

The current supplied to the D/A converters can be configured in the A_DAC_CTRL control register using the DAC_CTRL_CURRENT0 and DAC_CTRL_CURRENT1 bit fields. In general, if sampling frequencies below 20 kHz are desired, the nominal value (see below) should be used. For higher sampling frequencies, a higher current should be applied to prevent a reduction in signal fidelity.

The nominal current I_{nom} is 45 μ A if the pin AOR is attached to V_{REG} or 75 μ A if AOR is attached to ground, with both configurations operating at 1.28 MHz with a supply voltage of 1.25 V. AOR is the reference voltage for the analog components of the output stage and is connected to ground in most audio applications.

4.3.1 Control and Configuration Registers

Register Name	Register Description	Address
A_OUTPUT_CTRL	Output channel configuration	A:0x1E
A_DAC_CTRL	Select current to D/A converters	A:0x19

4.3.1.1 A_OUTPUT_CTRL Settings

Bit Field	Field Name	Field Description
3	OUTPUT_CTRL_DA1_ENABLE	Enable/disable D/A converter 1
2	OUTPUT_CTRL_DA0_ENABLE	Enable/disable D/A converter 0

Field Name	Value Symbol	Value Description	Hex Value
OUTPUT_CTRL_DA1_ENABLE	DA1_DISABLE*	Disable D/A converter 1	0x0
	DA1_ENABLE	Enable D/A converter 1	0x1
OUTPUT_CTRL_DA0_ENABLE	DA0_DISABLE	Disable D/A converter 0	0x0
	DA0_ENABLE*	Enable D/A converter 0	0x1

4.3.1.2 A_DAC_CTRL Settings

Bit Field	Field Name	Field Description
5:4	DAC_CTRL_CURRENT1	Select current for D/A converter 1
1:0	DAC_CTRL_CURRENT0	Select current for D/A converter 0

Field Name	Value Symbol	Value Description	Hex Value
DAC_CTRL_CURRENT1	DAC1_CURRENT_2_0X	Select 2 x Inom	0x0
	DAC1_CURRENT_1_0X*	Select Inom	0x1
	DAC1_CURRENT_0_7X	Select 0.7 x Inom	0x2
	DAC1_CURRENT_0_5X	Select 0.5 x Inom	0x3
DAC_CTRL_CURRENT0	DAC0_CURRENT_2_0X	Select 2 x Inom	0x0
	DAC0_CURRENT_1_0X*	Select Inom	0x1
	DAC0_CURRENT_0_7X	Select 0.7 x Inom	0x2
	DAC0_CURRENT_0_5X	Select 0.5 x Inom	0x3

4.4 OUTPUT RECONSTRUCTION FILTERS

The output reconstruction filters (shown as LPF0 and LPF1 on Figure 3 on page 18) when enabled, remove high-frequency aliasing components, as a result of digital sampling, from the outputs of both D/A converters. Both filters are based on a 3rd-order, active low-pass filter with a -3 dB cut-off frequency of 10 kHz or 20 kHz, which may be configured using the OUTPUT_CTRL_LPF_MODE_OUT bit of the A_OUTPUT_CTRL register.

4.4.1 Control and Configuration Registers

Register Name	Register Description	Address
A_OUTPUT_CTRL	Output channel configuration	A:0x1E

4.4.1.1 A_OUTPUT_CTRL Settings

Bit Field	Field Name	Field Description
6	OUTPUT_CTRL_LPF_MODE_OUT	Select the low-pass filter cut-off frequency

Field Name	Value Symbol	Value Description	Hex Value
OUTPUT_CTRL_LPF_MODE_OUT	LPF_OUT_ACTIVE_10KHZ*	Select 10 kHz output filtering	0x0
	LPF_OUT_ACTIVE_20KHZ	Select 20 kHz output filtering	0x1

4.5 OUTPUT ATTENUATORS

The output attenuators can reduce the analog output signal levels. This is useful when trying to match the signal output level to a given component's (e.g. a receiver's) dynamic range.

An attenuation value for each of the two attenuators is selected through the A_OUT_ATTEN_CTRL control register using the OUT_ATTEN_CTRL_OUT0_ATTEN and OUT_ATTEN_CTRL_OUT1_ATTEN bit fields.

The output signal on each analog output can be muted if desired. Output muting is performed in the A_OUT_ATTEN_CTRL control register using the OUT_ATTEN_CTRL_MUTE0 and OUT_ATTEN_CTRL_MUTE1 bit fields.

4.5.1 Control and Configuration Registers

Register Name	Register Description	Address
A_OUT_ATTEN_CTRL	Select attenuation values for the two attenuators	A:0x18

4.5.1.1 A_OUT_ATTEN_CTRL Settings

Bit Field	Field Name	Field Description
7:5	OUT_ATTEN_CTRL_OUT1_ATTEN	Select attenuation value for attenuator 1
4	OUT_ATTEN_CTRL_MUTE1	Mute/pass signal to analog output 1
3:1	OUT_ATTEN_CTRL_OUT0_ATTEN	Select attenuation value for attenuator 0
0	OUT_ATTEN_CTRL_MUTE0	Mute/pass signal to analog output 0

Field Name	Value Symbol	Value Description	Hex Value
OUT_ATTN_CTRL_OUT1_ATTN	OUTPUT1_ATTN_BYPASS*	Bypass attenuator	0x0
	OUTPUT1_ATTN_12DB	Set attenuator to 12 dB	0x1
	OUTPUT1_ATTN_15DB	Set attenuator to 15 dB	0x2
	OUTPUT1_ATTN_18DB	Set attenuator to 18 dB	0x3
	OUTPUT1_ATTN_21DB	Set attenuator to 21 dB	0x4
	OUTPUT1_ATTN_24DB	Set attenuator to 24 dB	0x5
	OUTPUT1_ATTN_27DB	Set attenuator to 27 dB	0x6
	OUTPUT1_ATTN_30DB	Set attenuator to 30 dB	0x7
OUT_ATTN_CTRL_OUT0_ATTN	OUTPUT0_ATTN_BYPASS*	Bypass attenuator	0x0
	OUTPUT0_ATTN_12DB	Set attenuator to 12 dB	0x1
	OUTPUT0_ATTN_15DB	Set attenuator to 15 dB	0x2
	OUTPUT0_ATTN_18DB	Set attenuator to 18 dB	0x3
	OUTPUT0_ATTN_21DB	Set attenuator to 21 dB	0x4
	OUTPUT0_ATTN_24DB	Set attenuator to 24 dB	0x5
	OUTPUT0_ATTN_27DB	Set attenuator to 27 dB	0x6
	OUTPUT0_ATTN_30DB	Set attenuator to 30 dB	0x7
OUT_ATTN_CTRL_MUTE1	OUTPUT1_PASS	Pass output signal	0x0
	OUTPUT1_MUTE*	Mute output signal	0x1
OUT_ATTN_CTRL_MUTE0	OUTPUT0_PASS	Pass output signal	0x0
	OUTPUT0_MUTE*	Mute output signal	0x1

4.6 DIRECT DIGITAL OUTPUT DRIVES

The purpose of the two on-chip direct digital outputs is to drive receivers or speakers without the need for an external amplifier.

The two sigma-delta type direct digital outputs are based on a pulse-density modulation (PDM) scheme and are enabled or disabled in the A_OUTPUT_CTRL control register using the OUTPUT_CTRL_OD0_ENABLE and OUTPUT_CTRL_OD1_ENABLE bits. Configuring the output mode using the OUTPUT_CTRL_HPWR_MODE bit in the A_OUTPUT_CTRL register indicates whether the direct digital outputs are to separately process the two output channels (stereo operation) or are to be linked together to produce a high-power output based on the data from channel zero (see Figure 4). When operating in high-power mode the OUTPUT_CTRL_OD1_ENABLE bit is ignored and output enabling is done only through the OUTPUT_CTRL_OD0_ENABLE control register field. It should be noted that the output pads used by direct digital output one are multiplexed with the two analog outputs. In the case where direct drive one is enabled or the direct drive outputs are configured for high-power mode, the output pads will be configured for digital outputs; otherwise the pads are configured for analog outputs.

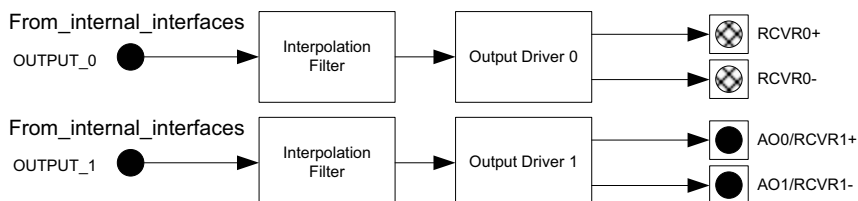
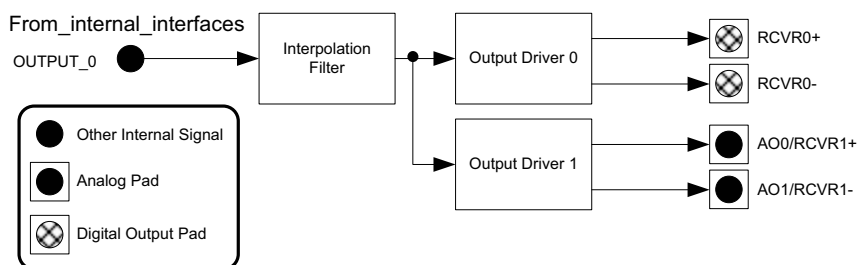
Regular Mode**High-Power Mode**

Figure 4: Internal channel assignments in regular and high-power modes

When configured for high-power mode, the synchronized output signals need to be combined. `RCVR0+` and `RCVR1+` both need to be connected to a single terminal on an output transducer, and `RCVR0-` and `RCVR1-` both need to be connected to the other terminal. When laying out the circuit for the application, these pins can be connected as shown in Figure 5.

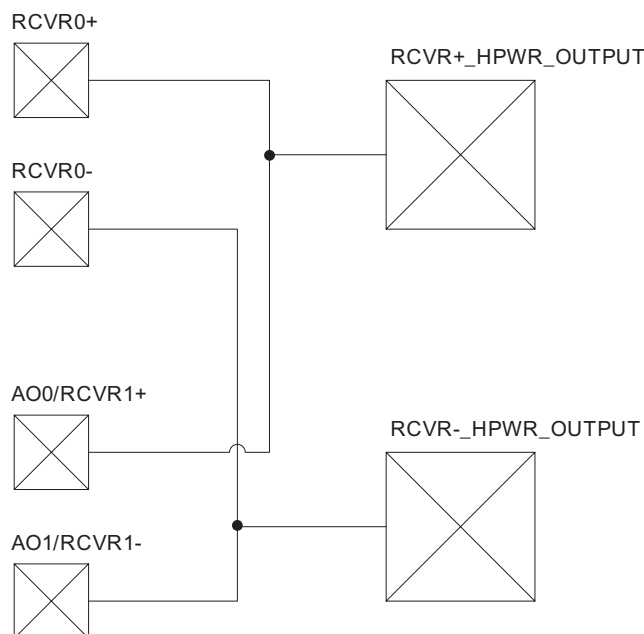


Figure 5: Signal routing of external connections for high-power output mode.

Note: The audio fidelity when in high-power mode can be slightly degraded compared to normal mode depending on the application. Prior to building a prototype with the configuration shown in Figure 5, we welcome you to contact Dspfactory for further implementation details.

When the 16-bit output signal from the IOP is passed to the direct digital outputs, it is oversampled, modulated, and quantized to one bit. This process results in a noise spectrum for the output signal that is no longer flat, but instead climbs for higher frequencies (this is referred to as *noise shaping*). For higher modulator sampling frequencies, the noise floor will start to climb at higher input frequencies. The modulator sampling frequency used in the direct digital output is configurable using the DAC_CTRL_OSR bit field in the A_DAC_CTRL control register. A total of four modulator sampling frequencies can be selected, each of which is a fraction of the frequency of MCLK: f_{MCLK} , $f_{MCLK}/2$, $f_{MCLK}/3$, $f_{MCLK}/4$.

For a detailed description of MCLK configuration and sampling frequency selection, see Section 9.3, “Oscillator Circuitry and Clock Generation” on page 107.

Figure 6 shows the effect on the output noise floor of varying the oversampling ratio (in this example the DAC sampling frequency is set to 16 kHz and the frequency of MCLK is 1.28 MHz).

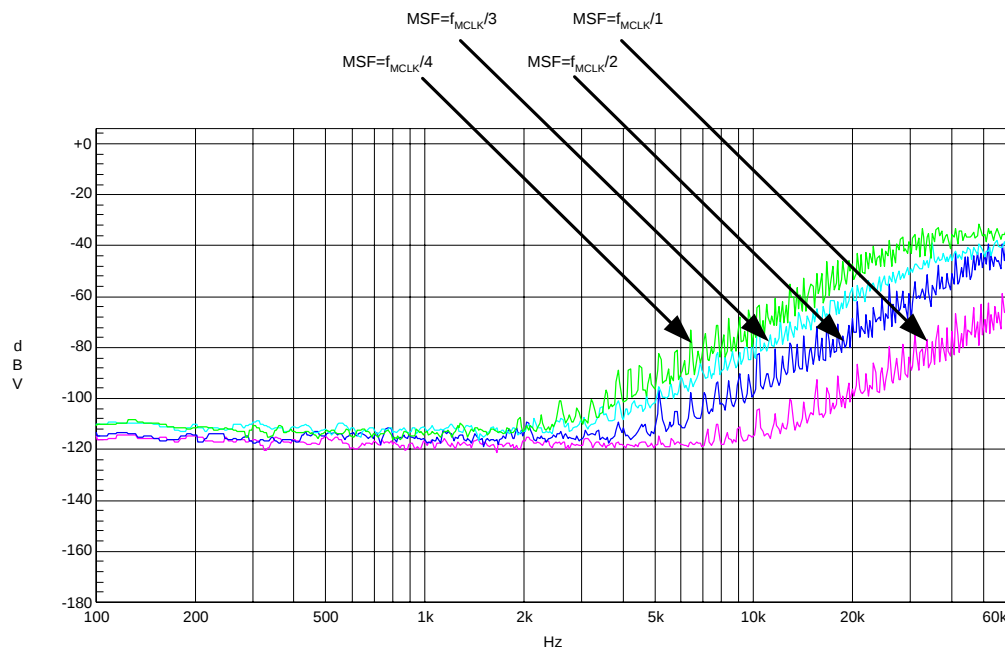


Figure 6: Noise floor showing the effect of oversampling rates of output signal. The sampling frequency is set to 16 kHz and the MCLK frequency is 1.28 MHz. MSF represents the output modulator sampling frequency.

4.6.1 Control and Configuration Registers

Register Name	Register Description	Address
A_OUTPUT_CTRL	Select direct digital output configuration	A:0x1E
A_DAC_CTRL	Direct digital output oversampling frequency selection	A:0x19

4.6.1.1 A_OUTPUT_CTRL Settings

Bit	Field Name	Field Description
7	OUTPUT_CTRL_HPWR_MODE	Enable/disable high-power output mode
1	OUTPUT_CTRL_OD1_ENABLE	Enable/disable direct digital output 1
0	OUTPUT_CTRL_OD0_ENABLE	Enable/disable direct digital output 0

Field Name	Value Symbol	Value Description	Hex Value
OUTPUT_CTRL_HPWR_MODE	OUTPUT_HIGHPOWER_DISABLE*	Select stereo output mode	0x0
	OUTPUT_HIGHPOWER_ENABLE	Select high-power output mode	0x1
OUTPUT_CTRL_OD1_ENABLE	OD1_DISABLE*	Disable direct digital output 1	0x0
	OD1_ENABLE	Enable direct digital output 1	0x1
OUTPUT_CTRL_OD0_ENABLE	OD0_DISABLE*	Disable direct digital output 0	0x0
	OD0_ENABLE	Enable direct digital output 0	0x1

4.6.1.2 A_DAC_CTRL Settings

Bit Field	Field Name	Field Description
7:6	DAC_CTRL_OSR	Select direct digital drive oversampling frequency

Field Name	Value Symbol	Value Description	Hex Value
DAC_CTRL_OSR	OSR_PRESCALE_1*	Select MCLK/1 modulator sampling frequency	0x0
	OSR_PRESCALE_2	Select MCLK/2 modulator sampling frequency	0x1
	OSR_PRESCALE_3	Select MCLK/3 modulator sampling frequency	0x2
	OSR_PRESCALE_4	Select MCLK/4 modulator sampling frequency	0x3

4.7 ANALOG OUTPUTS

The purpose of the analog outputs is to provide signals that can be used as inputs for external amplifiers or other components which can use an analog high-impedance input. If an external PWM class D amplifier is used, Orela 4500 can provide a clock synchronization signal. This signal is called USRCLK and will be explained in more detail in Section 9.3, "Oscillator Circuitry and Clock Generation" on page 107.

The analog outputs are multiplexed with the outputs of direct digital output one and are only enabled when direct digital output one and high-power mode are both disabled.

WOLA Filterbank Coprocessor

5.1 WOLA FILTERBANK COPROCESSOR

The Weighted Overlap-Add (WOLA) filterbank coprocessor implements an oversampled WOLA filterbank that operates on blocks of data. The WOLA filterbank provides a specific number of uniform-width signal-processing bands. Each band of the WOLA filterbank contributes to a time-frequency representation of the input signal, and can be read and processed by the RCore. The WOLA filterbank is highly configurable and introduces very low group delay. It can efficiently process input signals using from 4 to 128 frequency bands.

The WOLA configuration is set primarily using microcode and windows (easily loaded using a macro provided in the `<wola.inc>` include file). However, appropriate IOP configuration and WOLA control registers must also be set for the specific WOLA parameters as described below. The WOLA filterbank coprocessor is configured by setting the `D_IO_PROC_CFG` control register with the appropriate IOP configuration (for more information, see Chapter 6, “Input/Output Processor (IOP)” on page 49) and loading the appropriate microcode and window coefficients into predefined memory spaces (see Chapter 7, “Memory” on page 63).

5.1.1 WOLA Control Signals

Four primary signals are used to control and monitor the operation of the WOLA filterbank coprocessor. These signals are outlined in Table 1.

TABLE 1: WOLA CONTROL SIGNALS

Signal	Function
<code>SYS_CTRL_WOLA_START</code>	Setting this bit of the <code>D_SYS_CTRL</code> register to 1 starts an operation on the WOLA filterbank coprocessor as specified in the <code>SYS_CTRL_WOLA_FUNCTION</code> setting.
<code>SYS_CTRL_WOLA_RESET</code>	Setting this bit of the <code>D_SYS_CTRL</code> register to 1 resets all of the internal WOLA filterbank coprocessor states to zero.
<code>SYS_CTRL_WOLA_BUSY</code>	This signal at bit position <code>SYS_CTRL_WOLA_BUSY_POS</code> in the <code>D_SYS_CTRL</code> control register may be tested to determine whether the WOLA filterbank coprocessor is active (1) or idle (0).
<code>WOLA_DONE</code>	This interrupt is asserted when a WOLA operation completes.

The `SYS_CTRL_WOLA_FUNCTION` bit field in the `D_SYS_CTRL` register is composed of two bits that define the operation that the WOLA will perform when started. This operation is specific to the microcode that is loaded, but will be one of analysis, gain application, or synthesis. Table 2 lists the typical bit-field settings for `SYS_CTRL_WOLA_FUNCTION` in the `D_SYS_CTRL` register.

TABLE 2: TYPICAL BIT-FIELD SETTINGS FOR SYS_CTRL_WOLA_FUNCTION

#define	Bit 1	Bit 0	Function
UCODE_ANALYSIS_FUNCTION	0	0	Analysis
UCODE_GAIN_FUNCTION	0	1	Gain application
UCODE_SYNTHESIS_FUNCTION	1	0	Synthesis

The Basic Operating System (BOS) includes the macro `Start_WOLA(<FUNCTION>)` to control (configure and start) the WOLA coprocessor. For more information on using the `Start_WOLA` macro, see the *Orela 4500 Firmware Reference Manual*.

5.1.2 WOLA Operation

Depending on configuration, the WOLA filterbank typically performs three basic operations:

- Analysis – Time-domain data from the input FIFO are transformed into a frequency domain representation (complex output in rectangular format). Energy values may also be calculated.
- Gain Application – Real or complex gains are applied to the frequency domain data before synthesis. This is a fast vector multiply between a vector of gains and the transformed results of Analysis.
- Synthesis – Complex frequency domain data are transformed to the time domain.

The WOLA is capable of processing data in various different modes, which can be selected according to what is desired for a particular application. These modes include Mono, Simple Stereo, Full Stereo, and Digital Mixed.

Table 3 shows the WOLA processing modes and the corresponding FIFO configurations. Note that, in digital mixed mode, the RCore must perform any necessary channel combination on the input FIFO data before starting the analysis operation. Otherwise, the WOLA coprocessor will only perform a mono analysis on the first channel.

TABLE 3: WOLA PROCESSING MODES AND CORRESPONDING FIFO CONFIGURATIONS

WOLA Processing Mode	Input FIFO Configuration	Analysis Mode	Gain Application Mode	Synthesis Mode	Output FIFO Configuration
Mono	Input samples from a single channel are stored sequentially	Mono	Mono	Mono	One output channel is active and samples are stored sequentially
Simple Stereo	Input samples from two channels are stored interleaved	Stereo	Stereo	Mono	One output channel is active and samples are stored sequentially

TABLE 3: WOLA PROCESSING MODES AND CORRESPONDING FIFO CONFIGURATIONS (CONTINUED)

WOLA Processing Mode	Input FIFO Configuration	Analysis Mode	Gain Application Mode	Synthesis Mode	Output FIFO Configuration
Full Stereo	Input samples from two channels are stored interleaved	Stereo	Stereo	Stereo	Output samples of the two channels are stored interleaved
Digital Mixed	Input samples from two channels are stored in two separate blocks, where the samples from the first channel are stored sequentially in the first block, and the samples from the second channel are stored sequentially in the second block	Mono	Mono	Mono	One output channel is active and samples are stored sequentially

Note: To achieve a mono in/stereo out configuration, use Full Stereo mode and ignore one input.

A number of important parameters determine the operation of the WOLA filterbank coprocessor. Table 4 lists these parameters.

TABLE 4: PARAMETERS FOR THE WOLA FILTERBANK COPROCESSOR

Parameter	Description
Window Length (L)	The analysis window length is determined by the parameter L, which must be a power-of-two between 32 and 256. For stereo configuration, the maximum L is 128.
Window Coefficients	The window coefficients supplied by the EDK are typically loaded using <code><wola.inc></code> . It is also possible to load user-defined windows to the predefined memory spaces.
Input Block Size (R)	The input block size (R) represents the number of new input samples to be used for the current analysis. In other words, at each new block to be processed by the WOLA filterbank, R new samples are taken into account and appended to the previous L-R samples. R must be a power-of-two between 1 and L. <i>Note:</i> An additional parameter—defined as the oversampling (OS) factor—is sometimes used in the filterbank terminology. It is related to R by $OS=N/R$, where N is the FFT size.
FFT Size (N)	The FFT size determines how many bands are provided by the WOLA filterbank coprocessor. In even stacking there are $(N/2 - 1)$ full bands and two half bands. In odd stacking, there are $N/2$ full bands. The maximum value for N is 256 in mono mode, and 128 in stereo mode. <i>Note:</i> The WOLA coprocessor implements a filterbank, and not an FFT. An FFT block is used in the filterbank, but just as a component of the whole process.

TABLE 4: PARAMETERS FOR THE WOLA FILTERBANK COPROCESSOR (CONTINUED)

Parameter	Description
Decimation Factor (DF)	The decimation factor determines by how much the analysis window is decimated to generate the synthesis window. $DF = \frac{\text{length of analysis window}}{\text{length of synthesis window}}$ <i>Note:</i> The ratio between DF and OS (as well as the analysis window shape) has an influence on the scale of the signal produced by the WOLA after synthesis.
Stacking (even or odd)	Two settings are possible for the stacking: even or odd. Even stacking provides $(N/2 - 1)$ full bands with complex data and two half bands (located at DC and the Nyquist frequencies) with real data. Odd stacking provides $N/2$ bands all with complex data.

More detail on these operations, modes, and parameter settings is provided in the following sections.

5.1.3 WOLA Analysis

5.1.3.1 Mono Analysis

During analysis, the WOLA filterbank processes data from the input FIFO, R samples at a time. Data from the input FIFO are multiplied by a prototype analysis filter (W) with L coefficients. The windowed input data (L samples) are then “folded” and added. A circular shift is applied to the resulting buffer and an N -point FFT (in even or odd stacking) is computed. The sign sequencer ensures that the basis functions are continuous in odd stacking mode. Figure 7 shows a graphical representation of these operations.

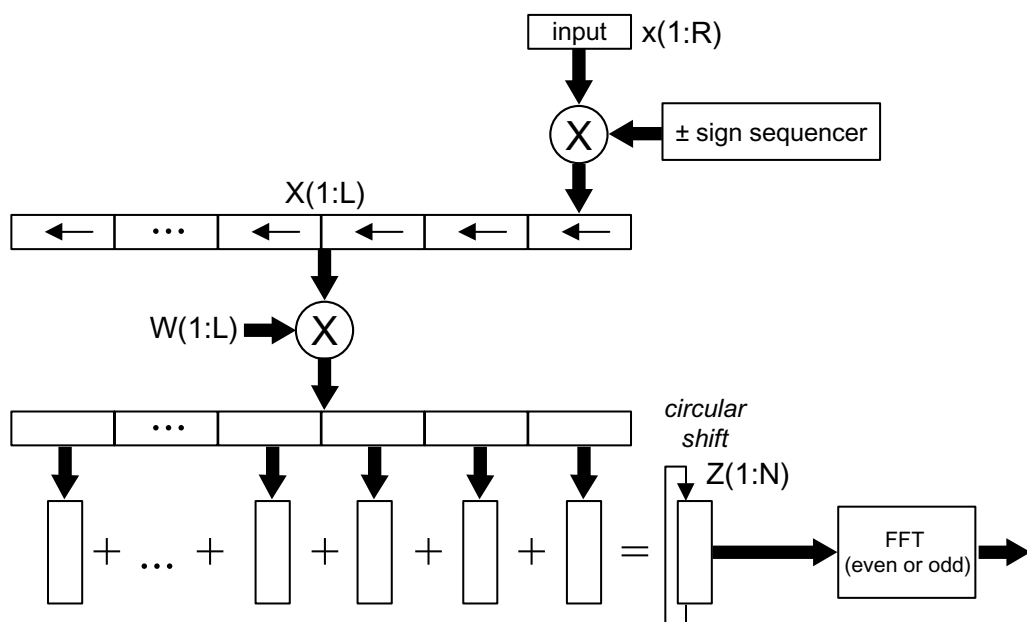


Figure 7: Analysis operation. N is the FFT size.

The results of the analysis operation are stored in X memory (see Chapter 7, “Memory” on page 63) starting at `D_WOLA_RESULT_BASE`. For odd stacking, the analysis operation generates a complex-valued result for each band (for a total of $N/2$ bands). For even stacking, the analysis operation generates $N/2+1$ bands. However, the DC and Nyquist bands are only half the normal sub-band bandwidth. Furthermore, in even stacking there is no imaginary component for the DC and Nyquist bands. Instead, the real component of the Nyquist band is packed into the memory location that is normally occupied by the imaginary component of the first band in odd stacking. Figure 8 illustrates the data organization of the analysis result in memory.

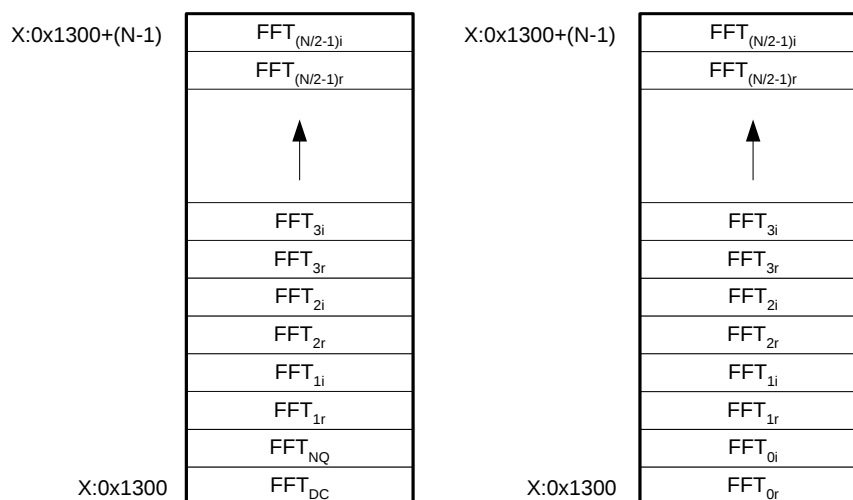


Figure 8: Mono analysis result. Left array shows data organization for even stacking. Right array shows data organization for odd stacking. N is the FFT size.

5.1.3.2 Stereo analysis

Orelia 4500 is capable of performing a stereo analysis operation within a single Analysis function call. This is accomplished by treating two real input data streams as a single complex data stream, where one input stream is treated as the real part and the other as the imaginary part.

As in mono analysis, the results of the stereo analysis are stored in X memory starting at `D_WOLA_RESULT_BASE`. Note that the organization of the analysis data is the same for both simple stereo and full stereo modes. Similar to the mono analysis, either even or odd stacking can be performed. Figure 9 shows the data organization of the analysis result after stereo analysis. X indicates the frequency-domain data for channel 0, and Y indicates the frequency-domain data for channel 1.

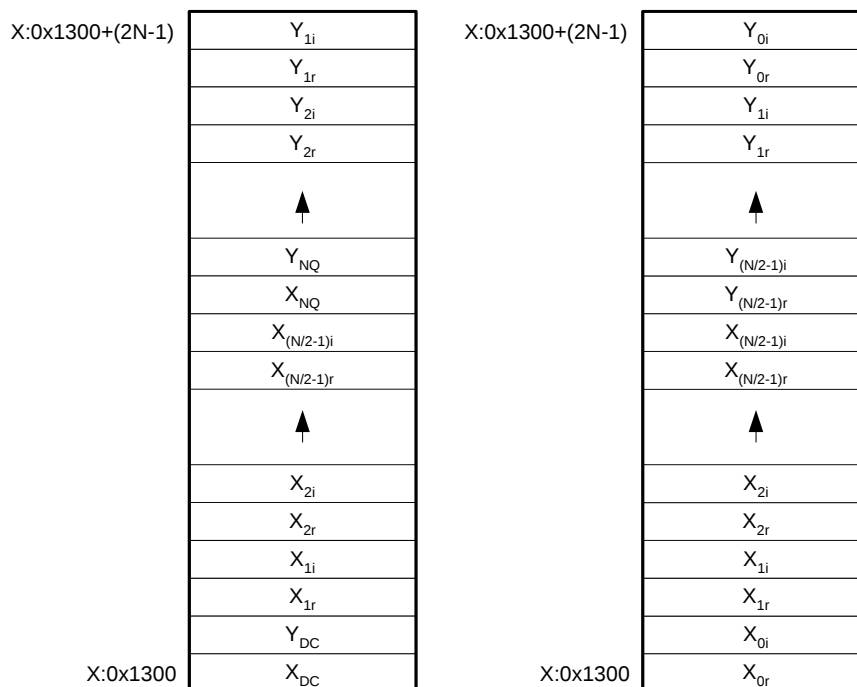


Figure 9: Stereo Analysis Results. The left array shows data organization for even stacking. The right array shows data organization for odd stacking. N is the FFT size.

The same data organization for the frequency domain data is used by the analysis, gain application and synthesis operations. The gain application and synthesis functions expect the data in X memory starting at `D_WOLA_RESULT_BASE` to be in the same format as the analysis result, except in the case of simple stereo mode, where the synthesis function treats the data as if a mono analysis has been performed.

5.1.4 WOLA Gain Application

The WOLA filterbank coprocessor can apply gains to the frequency domain data before the synthesis operation. The dedicated structure of the coprocessor is much more efficient than the structure of the RCore. As a result, using the coprocessor for gain application can lead to significant power savings. It also frees the RCore to perform other operations at the same time, maximizing parallel processing. The result of the gain application operation is stored in X memory starting at `D_WOLA_RESULT_BASE`, overwriting any value that these memory locations previously held.

5.1.4.1 Mono Gain Application

For mono gain application, the gains are stored in X memory starting at address `D_GAIN_BASE` using the format shown in Figure 10 on page 33. Within each band, the same gain is applied to the real and imaginary components; however, it is not necessary to write the gain twice except for the first band. In odd stacking, the gain for the first band must be written twice, while in even stacking, the gain for the Nyquist band is written to the same location where the imaginary data

component is stored. After the appropriate gains are stored in the memory, the gain application operation can be started by the `Start_WOLA()` macro.

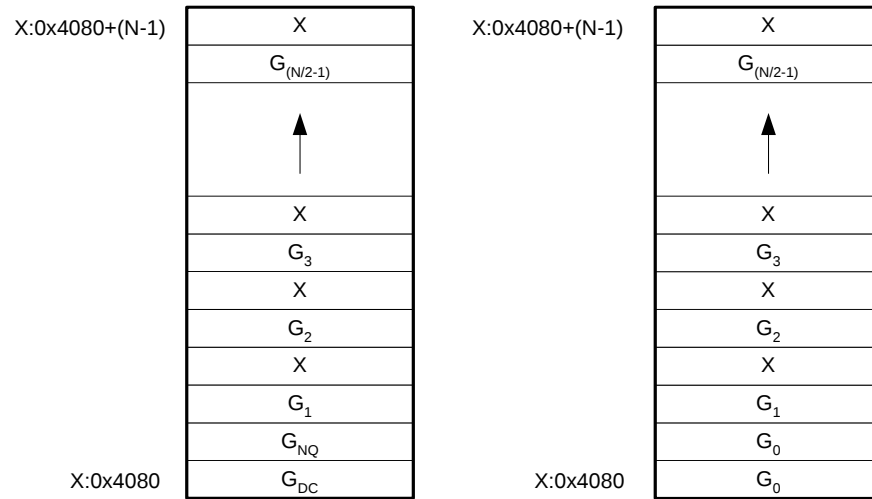


Figure 10: Mono real gain storage. The left array shows data organization for even stacking. The right array shows data organization for odd stacking. 'X' means "don't care". N is the FFT size.

5.1.4.2 Stereo Gain Application

In stereo mode, gains are applied to the two channels separately. Similar to mono mode, the gains are stored in X memory starting at D_GAIN_BASE (X:0x4080). However, the gains for the two channels are stored in two separate blocks as shown in Figure 11.

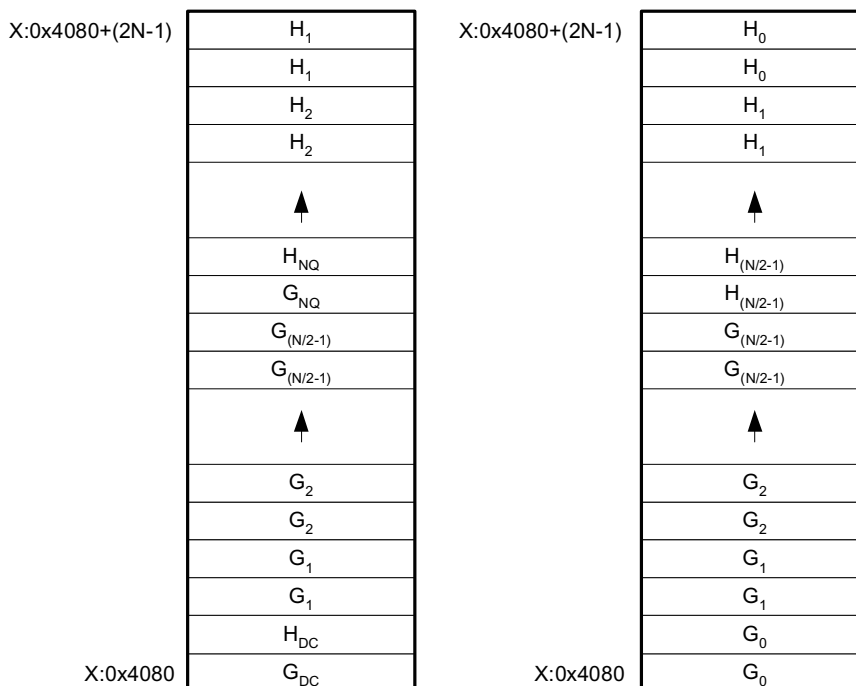


Figure 11: Stereo real gain storage. The left array shows data organization for even stacking. The right array shows data organization for odd stacking. 'X' means "don't care". 'G' indicates gains for channel zero, 'H' indicates gains for channel one. N is the FFT size.

In even stacking, the gains for the DC component of the two channels are stored in the first two memory locations of the first channel, while the gains for the Nyquist component are stored in the last two memory locations of the second channel.

Note: The order of the gain factors in the second channel is reversed in memory.

In simple stereo mode, the gain application operation also performs channel combination in addition to gain multiplication. The channel combination operation simply averages the two channels together after applying the gains. The result of the gain application operation is then stored in the same manner as in mono mode (i.e. in the X memory locations starting from D_WOLA_RESULT_BASE that previously held the analysis result for the first channel).

In full stereo mode, no channel combination takes place, and the results are written out to the X memory locations starting from D_WOLA_RESULT_BASE for the two channels.

5.1.4.3 Complex Gain Application

Complex gain application can be useful to manipulate the phase and magnitude information of the subband signals independently. On Orela 4500, the WOLA filterbank coprocessor can also perform complex gain application, which allows gains that are complex-valued to be multiplied with the complex frequency-domain data. Complex multiplication is used, which consists of four real multiplications, one addition, and one subtraction. This complex gain application operation is available to all mono and stereo processing modes in even or odd stacking. As with real gain application, the complex gains are also stored in X memory starting at D_GAIN_BASE. Furthermore, the memory locations that were specified as “don't care” in real gain application are now used to store the imaginary part of the gain factors. The result of the complex gain application operation is stored in the same manner as in the real-valued counterpart.

The following sections show the memory organization of the real and imaginary parts of the complex gains for each processing and stacking mode.

5.1.4.3.1 Mono Complex Gain Application

Figure 12 shows the data organization of the complex gain factors for mono mode.

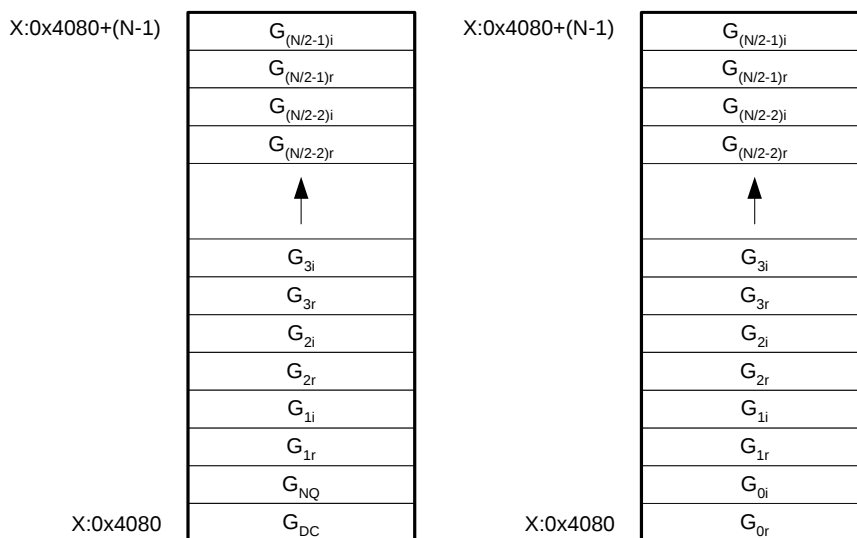


Figure 12: Complex gains for mono mode. The left array shows data organization for even stacking. The right array shows data organization for odd stacking. N is the FFT size.

5.1.4.3.2 Stereo Complex Gain Application

Figure 13 shows the data organization of the complex gain factors for stereo mode (simple stereo and full stereo).

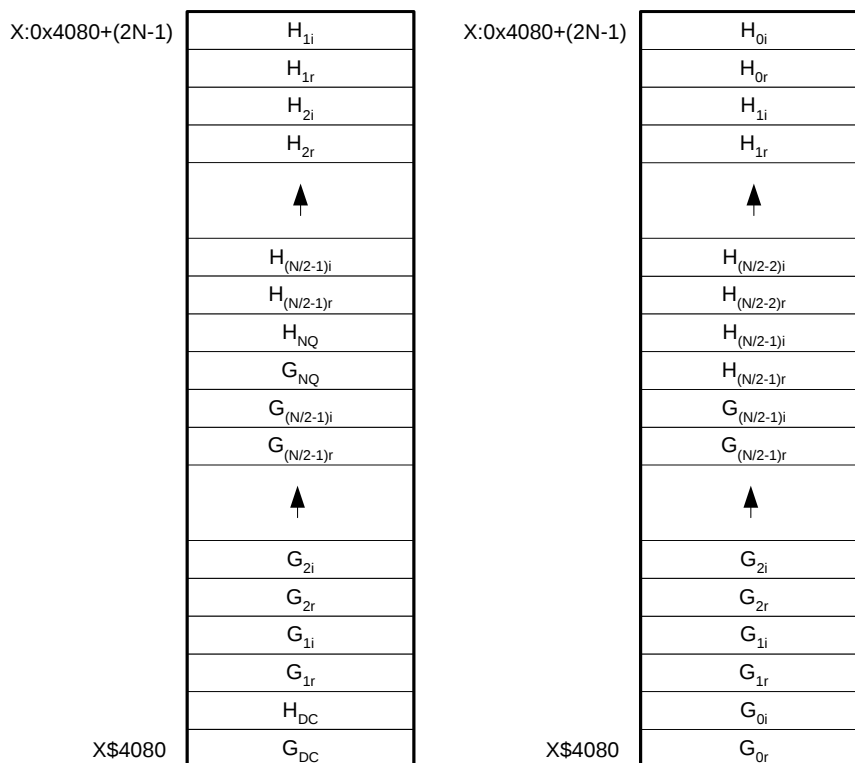


Figure 13: Complex gains for stereo mode (simple stereo and full stereo). The left array shows data organization for even stacking. The right array shows data organization for odd stacking. 'G' indicates gains for channel zero, 'H' indicates gains for channel one. N is the FFT size.

5.1.5 Synthesis

The synthesis portion of the coprocessor performs an inverse transform to the analysis processing (see Figure 14 on page 37). The frequency domain data stored in X memory starting at D_WOLA_RESULT_BASE are processed by an inverse FFT (in even or odd stacking). The output of the IFFT is then circularly shifted, periodically extended, and multiplied by the synthesis filter (which may be a decimated version of the analysis filter). The output samples are then reconstructed using an overlap-add technique and supplied to the output FIFO.

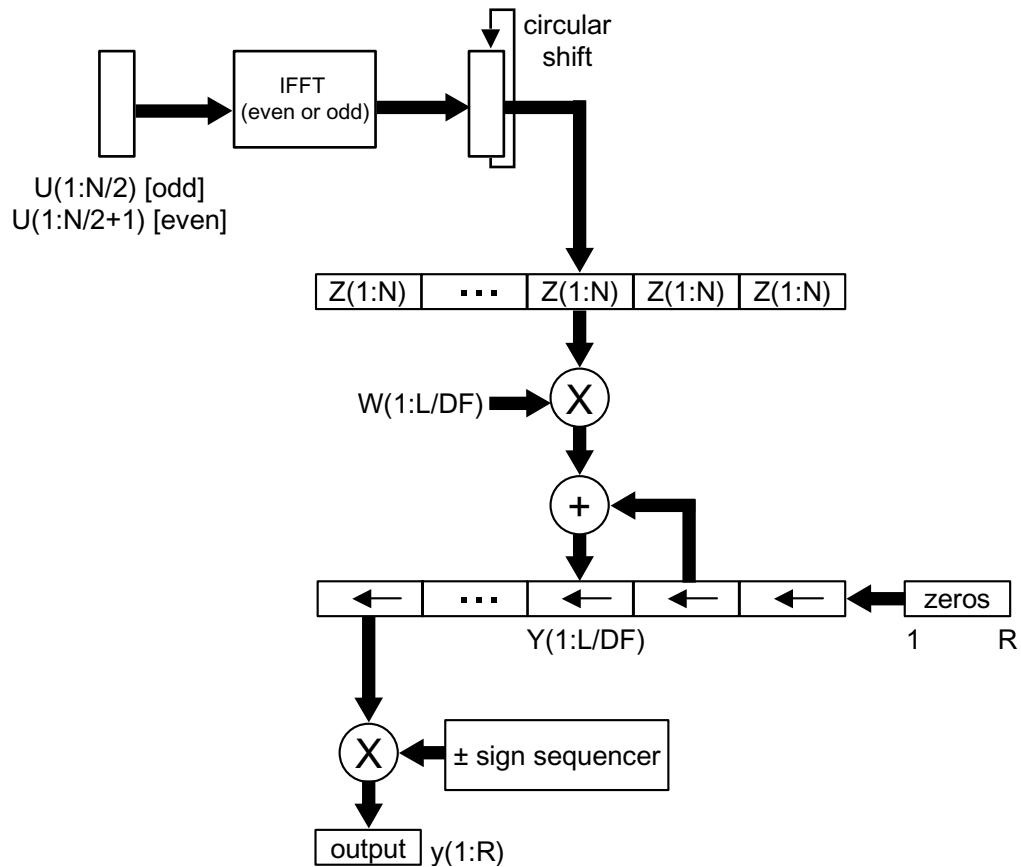


Figure 14: Synthesis operation. N is the FFT size.

On Orela 4500, as with the stereo analysis operation, a stereo synthesis operation is possible through the use of complex IFFT. The output samples of the two channels are reconstructed and written to the output FIFO.

5.1.6 Block Floating Point

To maintain numerical accuracy of the filterbank results, block floating point arithmetic and extended precision are used for all critical calculations. In fact, calculation steps in the filterbank are internally organized in several *passes*, in which numerical precision is 18-bit. At the end of each pass, data is rescaled to 16 bits, and the successive scaling factors involved are accumulated into the `D_BLOCK_EXP_DATA` register, as a number of right shifts. The `D_BLOCK_EXP_DATA` register is the block exponent register used by the WOLA filterbank coprocessor. Typically, at the end of the analysis, this register is written with the number of right shifts that were used to scale the data and avoid overflow during analysis. This value must be taken into account for any level calculations that are made on frequency domain output from the analysis filterbank. Basically, the frequency-domain results stored at location `D_WOLA_RESULT_BASE` have to be left-shifted again by the number of bits stored in `D_BLOCK_EXP_DATA`. However, access to the 18-bit precision results is offered to the user if necessary. Figure 15 shows how the results are "mirrored" so that all 18 bits can be read (if required).

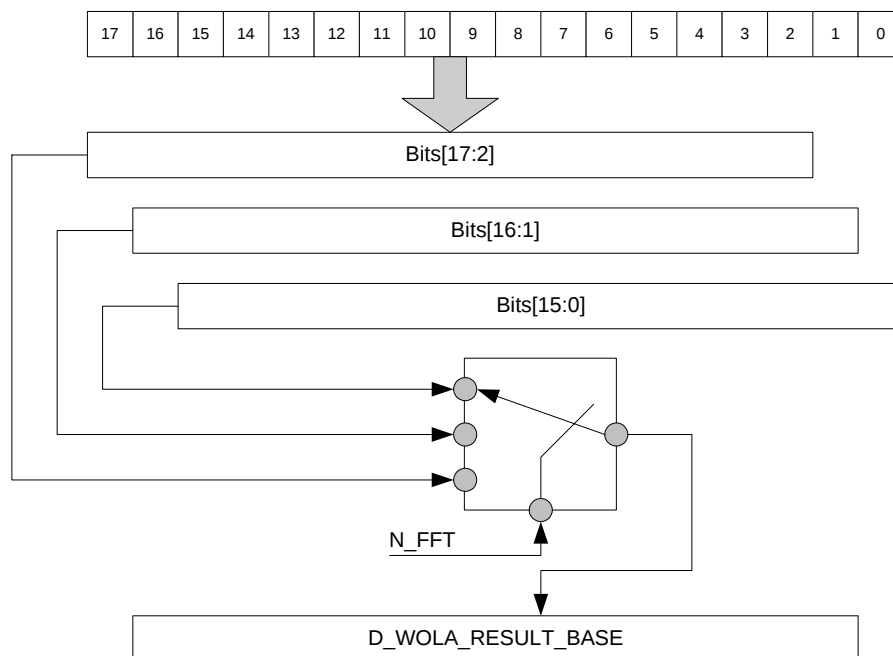


Figure 15: 18-bit frequency domain results.

In Figure 15, the `N_FFT` bits represent the number of shifts (0, 1, or 2) that are pending after the final pass of the analysis, which can be read by masking the value of the `D_MAGIC_READ` register using `MAGIC_READ_N_WOLA_MASK`. Reading the data starting at `D_WOLA_RESULT_BASE` means that the results are automatically scaled to incorporate this last shift. During synthesis, the coprocessor continues to accumulate the number of right shifts in `D_BLOCK_EXP_DATA` that were used to scale the data and avoid overflow. The time domain data are automatically scaled down by `FFT_SIZE` and scaled up by the final number of shifts after the synthesis operation completes. The aggregate number of shifts can be positive or negative.

Figure 19 shows a block diagram of the operation of the `D_BLOCK_EXP_DATA` and `D_GAIN_EXP_DATA` registers in conjunction with the analysis, gain application, and synthesis portions of the WOLA filterbank coprocessor.

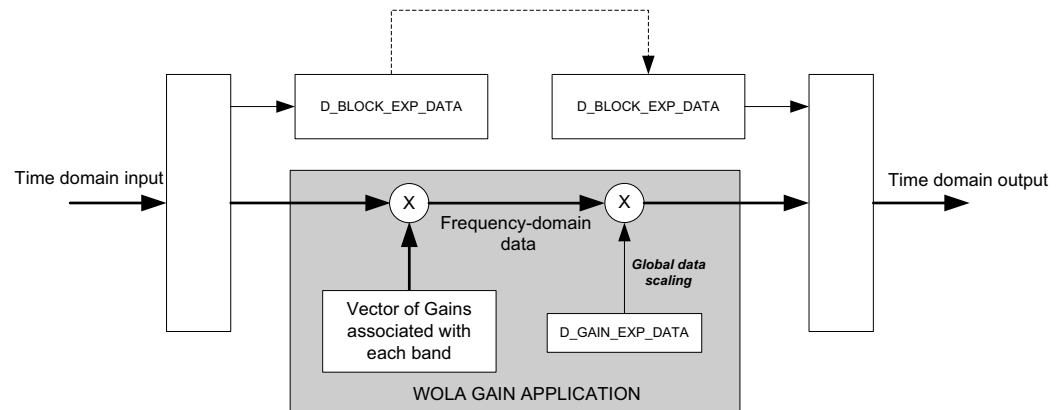


Figure 16: Block floating point operation used in the WOLA filterbank coprocessor.

The `D_GAIN_EXP_DATA` register allows the coprocessor to apply gains that are greater than one. It represents the number of left shifts (doublings) to be applied to the frequency domain results (expressed in the 16-bit fractional format) after multiplication by the gains located at `D_GAIN_BASE`. This feature must be used with caution because the results will be incorrect if the 18-bit registers overflow (even though saturation would be applied). This feature is intended for use with large dynamic range spectra—it can allow the lowest level portions of the spectrum to be amplified without sacrificing signal-to-noise ratio.

On Orela 4500, the `D_GAIN_EXP_DATA` register contains a 4-bit signed value, allowing up to 90 dB of digital gain to be applied in the gain application operation.

5.2 WOLA ENERGY CALCULATIONS

The WOLA supports the calculation of band energies, where the energy E for a specific band is defined as described in the equation below:

$$E = FFT_r^2 + FFT_i^2$$

The energy calculation functionality can be considered a part of the WOLA analysis function and therefore does not have a separate WOLA function. When starting the WOLA analysis function using microcode suitable for energy calculations, the WOLA filterbank coprocessor will first calculate the complex FFT points followed by the energy values.

The energy value for each band is stored in two consecutive memory cells in the Mirrored Temp memory range, as described in Section 7.1, “Architecture” on page 63. This is because the addition of the squared real and imaginary parts of a complex FFT-point results in a 32-bit number of the format (2.30), since the two values FFT_r and FFT_i each have the format (1.15). The format (2.30) indicates that a value has two integer bits (including sign-bit) and 30 fractional bits. Similarly, the format (1.15) indicates one sign-bit and 15 fractional bits.

Typically, the WOLA analysis results are accessed using the Mirrored Temp memory starting at address `X:0x1300`. Accessing analysis results in the memory range starting at this address causes the results to be shifted by the proper N_{FFT} factor upon access. However, when accessing the

energy values this access scheme is not valid since the `N_FFT` factor does not apply directly to the energy values (because one value is stored in two memory cells). The shifted access scheme applies only to the complex FFT points that are a result of the WOLA analysis process. Therefore, energy values must be accessed in the Mirrored Temp memory range that starts at address `X:0x1000`, since no scaling factor is being applied to values in this range upon access. Complex FFT points (that are scaled by the `N_FFT` factor) can still be accessed in the address range starting at `X:0x1300`.

The `D_BLOCK_EXP_DATA` factor is being used by the WOLA filterbank coprocessor to scale the complex FFT points to avoid overflow during the WOLA analysis. This factor is normally re-applied to the complex FFT-points during the synthesis. In a similar fashion a scaling factor must be applied to the energy values. This scaling factor (called `BLOCK_EXP_ENERGY`) is not calculated by the WOLA filterbank coprocessor, but must be calculated and applied by the RCore using the following formula:

$$BLOCK_EXP_ENERGY = (D_BLOCK_EXP_DATA) \cdot 2$$

The complex FFT points and the energy values physically share the same memory instance. Thus, the size of this memory instance determines how many complex FFT points and how many energy values can be stored simultaneously. Since this memory instance only has 256 memory cells, only complex FFT points and energy values can be present simultaneously for FFT sizes up to 64 for mono mode and for FFT sizes up to 32 for stereo mode. Calculating energy values for larger FFT sizes will result in complex FFT points being overwritten by energy values.

5.2.1 Mono Energy Calculations

Figure 17 shows the data organization of the energy values for mono mode.

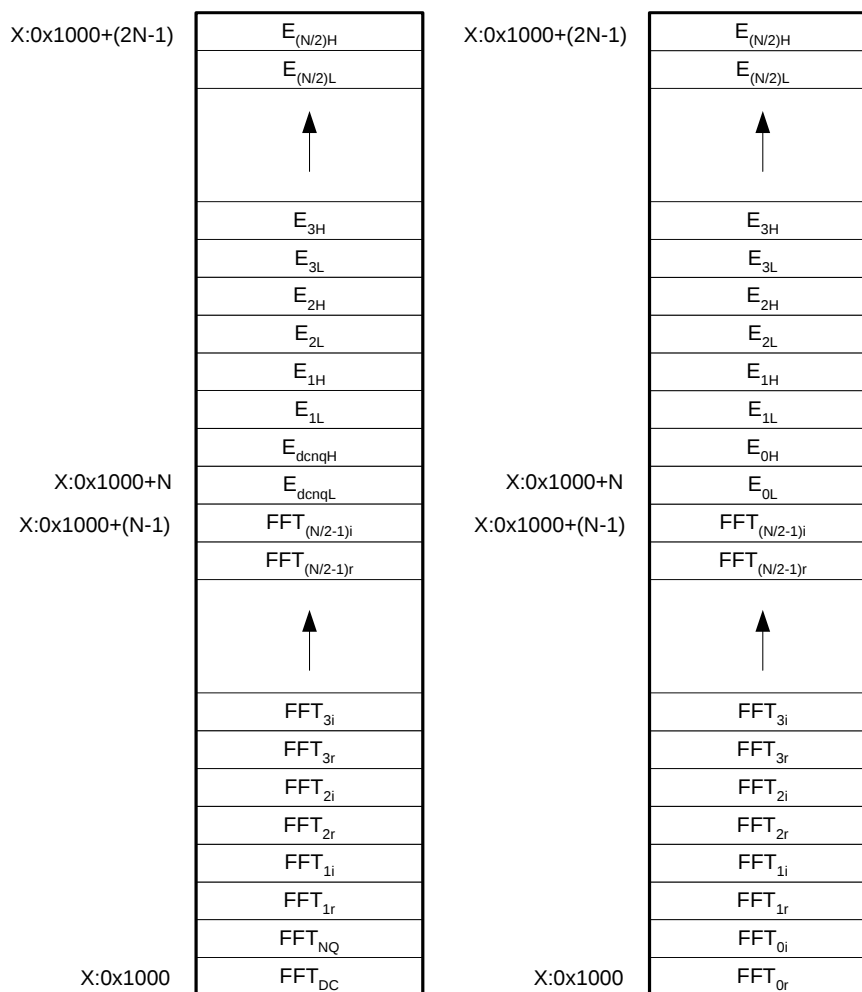


Figure 17: Energy calculations for mono mode. The left array shows data organization for even stacking. The right array shows data organization for odd stacking. N is the FFT size.

Note: In even stacking mode, the calculated energy value for the DC and Nyquist point (represented by E_{dcnqL} and E_{dcnqH}) is invalid since it is being calculated by squaring and adding the real FFT points related to the DC and Nyquist points respectively. If the energies for these bands are required, separate squaring of the two real values representing these bands is simple but must be performed in the RCore.

5.2.2 Stereo Energy Calculations

Figure 18 shows the data organization of the energy values for stereo mode.

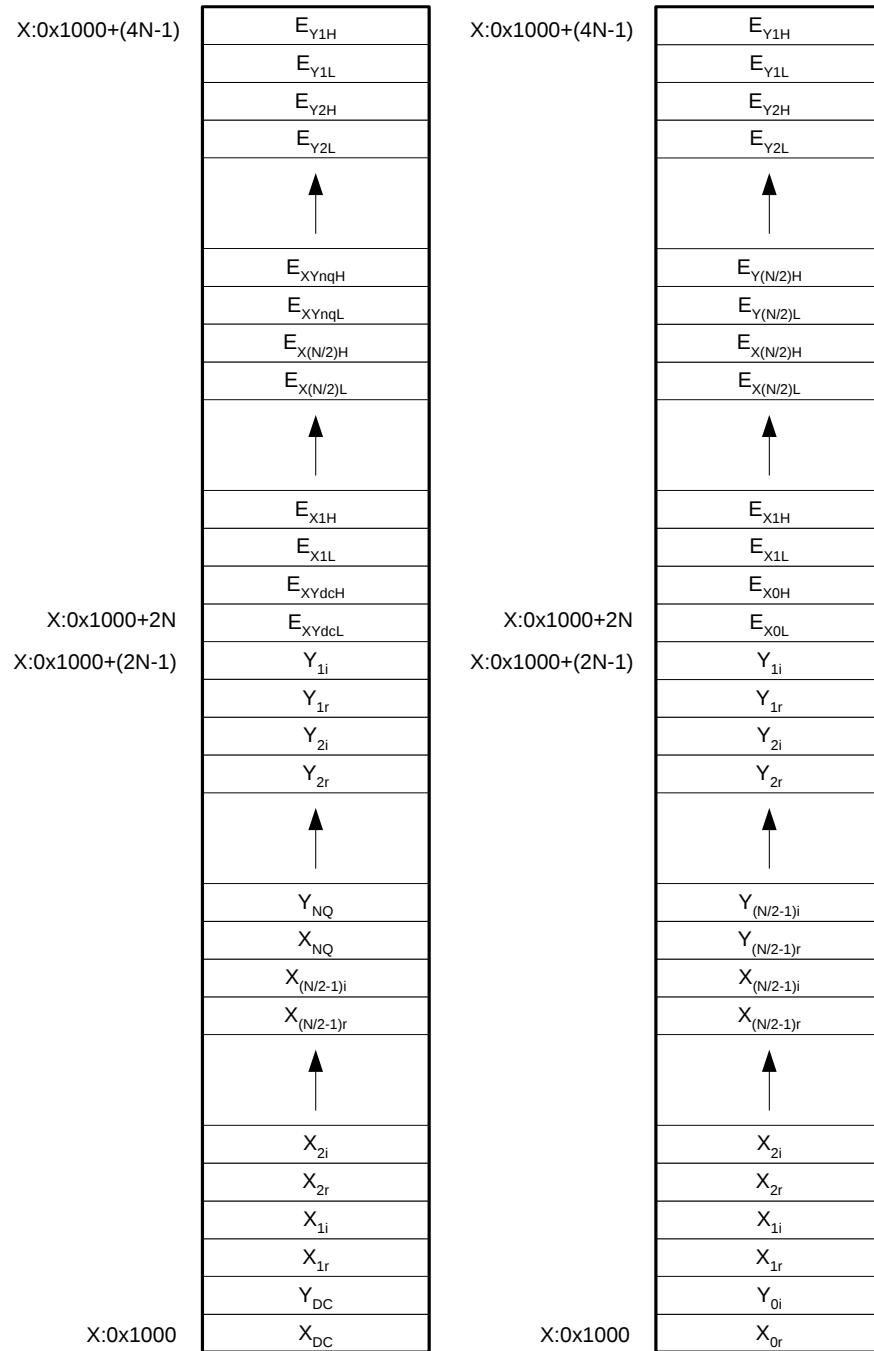


Figure 18: Energy calculations for stereo mode. The left array shows data organization for even stacking. The right array shows data organization for odd stacking. N is the FFT size.

Note: In even stacking mode, the calculated energy values for the DC and Nyquist points for each channel (represented by E_{XYnqL} , E_{XYnqH} , E_{XYdcL} and E_{XYdcH} .) are invalid since they are being calculated by squaring and adding the real FFT points related to the DC and Nyquist points respectively. If the energies for these bands are required, separate squaring of the two real values representing these bands must be performed in the RCore for each channel.

5.3 WOLA CONFIGURATION

As mentioned in Section 5.1, “WOLA Filterbank Coprocessor” on page 27, the WOLA filterbank coprocessor is configurable by setting the D_IO_PROC_CFG register and loading microcode and windows into predefined memory locations.

Table 5 lists the WOLA parameters that can be adjusted. In addition to the listed parameters, it must be noted that the window coefficients can also be freely chosen and stored in the dedicated locations. However, this should only be performed in special cases, and the choice of such customized windows should be carefully done. In most cases, reliance on Dspfactory's default windows, which are included with the microcode, is highly recommended. Those default windows have been extensively studied and optimized for most filterbank configurations.

TABLE 5: CONFIGURABLE PARAMETERS FOR WOLA

Parameter	Description
N	Number of points in FFT
L	Analysis window length per channel
R	Input block size per channel
DF	Decimation factor
OS	Oversampling factor (equal to N/R)
Mode	Mono, simple stereo, full stereo, or digital mixed
Gain Mode	Real or complex gain
Stacking	Even or odd

5.3.1 Include File

The macro WOLA_CONFIGURE() in the include file <wola.inc> is used to load the correct window and microcode files that are supplied with the evaluation kit. The parameters used in the macro are defined in Table 6.

WOLA_CONFIGURE(LEN, NFFT, OS, DF, MODE, STACKING)

TABLE 6: WOLA_CONFIGURE MACRO PARAMETERS

Parameter	Purpose
LEN	Length of the analysis window per channel (32, 64, 128 or 256). Stereo mode supports only up to L=128.
NFFT	FFT size (8 up to LEN in powers of two)
OS	Oversampling factor ($NFFT/R$, $1 \leq R \leq [NFFT \text{ or } 128 \text{ maximum}]$ and in powers of two)
DF	Decimation factor (1 up to OS in powers of two)
MODE	Audio processing mode (WOLA_MODE_STEREO, WOLA_MODE_MONO or WOLA_MODE_FULL_STEREO)
STACKING	WOLA_STACKING_EVEN or WOLA_STACKING_ODD

After the appropriate parameters have been set, the macro determines the appropriate microcode and windows to be loaded into predefined memory locations. The location for the microcode is in X memory starting at D_MICRO_BASE. The location for the analysis window is in X memory starting at D_ANA_WIN_BASE, and the location for the synthesis window is in X memory starting at D_SYN_WIN_BASE. For more information on using the WOLA_CONFIGURE macro see the *Orela 4500 Firmware Reference Manual*.

5.3.2 Input Block Size (R)

The input block size (R) determines the number of consecutive time-domain digital audio samples to be used by the WOLA as one block. The configuration of the input block size is described in detail in Chapter 6, "Input/Output Processor (IOP)" on page 49. Notice that in stereo modes, the block size parameter to be used by the IOP should be chosen as twice the value of R, as used by the WOLA. Effectively, the parameter R, in the WOLA sense, only concerns one channel. In contrast, in the IOP configuration, the block size configuration includes both channels.

5.3.3 Window Length (L)

The analysis window length (L) defines the size of the window for each of channel 0 and 1. The configuration of the window length is described in detail in Chapter 6, "Input/Output Processor (IOP)" on page 49. Notice that in stereo modes, the window length parameter to be used by the IOP should be chosen as twice the value of L, as used by the WOLA. Effectively, the parameter L, in the WOLA sense, only concerns one channel. In contrast, in the IOP configuration, the window size configuration includes both channels.

5.3.4 WOLA Performance

The WOLA parameter settings can affect a number of performance characteristics:

- The number of filter bands ($N/2$) and hence the width of each band.
- The filter cut-off characteristics (window)
- The center-frequency of each band (even/odd stacking)
- Block rate and number of cycles per block (R)
- Group delay of filterbank

- The level of aliasing distortion
- Relative power consumption

5.3.5 Block Rate (Tick)

The block rate (tick) is the time between two consecutive IO_BLOCK_FULL interrupts (enabled with INT_EBL_IO_BLOCK_FULL). The period of a tick is equal to the input block size (R) times the nominal sampling period (typically 1/(16 kHz)). The block rate determines the number of cycles available on the RCore between two consecutive IO_BLOCK_FULL interrupts. The number of cycles available in a tick is equal to the tick period times the SYS_CLK frequency. This is an important parameter because between each successive tick in a frequency-domain algorithm, the WOLA coprocessor must complete the synthesis operation to preserve the continuity of the output signal.

5.3.6 Delay

The WOLA filterbank has the following sources of delay:

- The Block Pipelining
- The Analysis Window
- The Synthesis Window

The analysis and synthesis windows are finite impulse response (FIR) filter structures, which introduce an algorithmic delay equal to the sum of half the window lengths. Thus, the analysis window introduces a delay equal to L/2 samples, while the synthesis window additionally introduces a delay equal to L/2DF. Including the block pipelining delay (R samples), the total group delay through the WOLA filterbank coprocessor can be found using the formula:

$$GroupDelay = \left(\frac{L}{2} + \frac{L}{2DF} + R \right) \cdot \frac{1}{F_s}$$

where L is the window length, DF is the decimation factor, R is the input block size and Fs is the sampling frequency. The group delay can be decreased by performing one or more of the following tasks:

- Reducing the window length (L)
- Increasing the decimation factor (DF)
- Increasing the oversampling rate (decreasing R)
- Increasing the nominal sampling rate (Fs)

5.3.7 Frequency Response

The parameters used to produce the frequency response as shown in Figure 19 are:

- System clock (SYS_CLK) at 1.28 MHz
- Nominal sampling frequency of 16 kHz
- 32-point FFT (N=32, 16 bands)
- 128-point analysis window (L=128)

- 32-point synthesis window (DF=4)
- 8-point input block size (R=8)

These parameters correspond to 4-times oversampling ($OS = N/R = 4$) and a 5.5 ms group delay. Figure 19 shows the frequency response for each band.

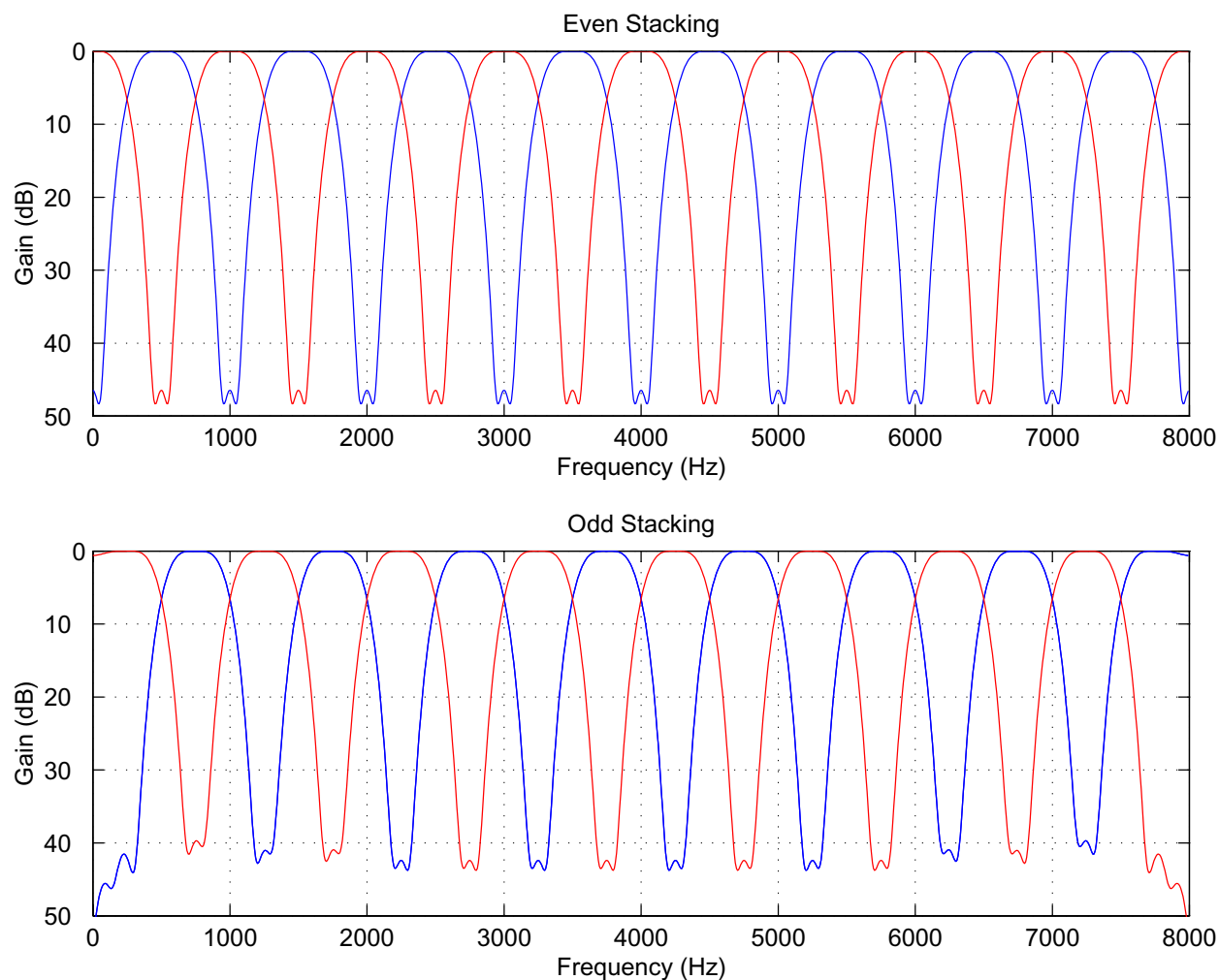


Figure 19: Frequency response for even- and odd-stacking bands.

5.3.7.1 Filterbank Aliasing Distortion

The level of aliasing distortion in the filterbank can be reduced by increasing the window length (L) at analysis and/or synthesis, or by increasing the oversampling rate OS (which corresponds to decreasing R if N is to be kept constant).

5.3.8 Power Consumption

The relative power consumption of the WOLA filterbank coprocessor depends on the block size, the window size, and the number of bands. Keeping N constant, the relative power consumption of the filterbank can be reduced by decreasing the window length at analysis and/or synthesis, or by decreasing the oversampling rate (increasing R).

This page left blank for printing purposes.

Input/Output Processor (IOP)

6.1 INTRODUCTION

The Input/Output Processor (IOP) controls the flow of audio samples from the input stage to the input FIFO where they can be accessed by the RCore/WOLA and from the output FIFO (where processed samples are left by the RCore/WOLA) to the output stage. Data transfers between the FIFOs and the input and output stages happen through shared memories and are synchronized using interrupts.

The IOP can be configured to operate in one of four different modes: mono, simple stereo, full stereo, digital mixed mode. Each mode couples tightly to the configuration of the input and output FIFOs and to that of the WOLA configuration (if used). The IOP is enabled/disabled and configured in the D_IO_PROC_CFG control register using the IO_PROC_CFG_ENABLE bit field and the IO_PROC_CFG_INPUT_STEREO, IO_PROC_CFG_OUTPUT_STEREO, IO_PROC_CFG_CHAN_SEP bit fields. Table 7 summarizes the operation modes and how to obtain them by configuring the bits in the D_IO_PROC_CFG register (for complete configuration information see Section 6.8, “Control and Configuration Registers” on page 61).

TABLE 7: SUMMARY OF IOP OPERATION MODES

IOP Processing Mode	D_IO_PROC_CFG settings
Mono	IOP_INPUT_MONO selected IOP_OUTPUT_MONO selected IOP_CHAN_INTERLEAVED selected
Simple Stereo	IOP_INPUT_STEREO selected IOP_OUTPUT_MONO selected IOP_CHAN_INTERLEAVED selected
Full Stereo	IOP_INPUT_STEREO selected IOP_OUTPUT_STEREO selected IOP_CHAN_INTERLEAVED selected
Digital Mixed	IOP_INPUT_STEREO selected IOP_OUTPUT_MONO selected IOP_CHAN_SEPARATED selected

The RCore can access samples in the input and output FIFOs either by addressing samples through the normal FIFO address range or through the Smart FIFO addressing scheme. In the normal FIFO, the address of the most recent input block shifts as new blocks of samples arrive. In the Smart FIFOs, the address of the most recent input and output blocks are fixed at a specific location. This is convenient if input samples need to be accessed before they are processed in the WOLA filterbank coprocessor.

Note: Access to the FIFOs is restricted due to the limited number of ports on the memories. Information regarding when a given memory may be accessed is presented in Chapter 7, “Memory” on page 63.

The IOP works on a number of domains in the FIFOs, as follows:

Input Block	The most recent input block is defined as the newest, stable block of input samples that can be accessed by the WOLA/RCore in the input FIFO.
Output Block	Like the input block, the most recent output block is defined as the block of samples that are currently being written to the output FIFO by the WOLA/RCore.
Write-In	The write-in domain is defined as the part of the input FIFO where new input samples are in the process of being stored by the IOP. When the number of samples in the write-in domain reaches the input block size (R) the write-in domain becomes the new, most recent input block and an <code>IO_BLOCK_FULL</code> interrupt is generated. The input block size (R) determines the number of consecutive time-domain digital audio samples per channel to be gathered by the IOP as one block. In stereo and digital mixed mode, the total block size (i.e. the total number of samples from both channels 0 and 1) is $2R$, which means that each block contains R samples from each channel. The 3-bit field <code>IO_PROC_CFG_BLK</code> in the <code>D_IO_PROC_CFG</code> control register determines R. (Note that the value specified here should equal the total number of blocks transferred for both channels; thus in stereo mode, if a value of $R=8$ per channel is desired, the block size in <code>IO_PROC_CFG_BLK</code> must be defined as 16.) R must be smaller than or equal to L.
Read-Out	Like the write-in domain, the read-out domain is defined as the part of the output FIFO where output samples are in the process of being read by the IOP. When the IOP has finished reading samples in the current read-out domain the current most recent output block becomes the new read-out domain.
Input Processing	The input processing domain is the set of L samples, where L is the processing window size per channel, being used by the WOLA filterbank coprocessor to perform analysis. In stereo and digital mixed mode, similar to the input block size, the total size of the window for both channel 0 and 1 together is $2L$ because the window length for each channel is L. The 2-bit field <code>IO_PROC_CFG_WIN</code> in the <code>D_IO_PROC_CFG</code> control register determines L. (Note that the value specified here should equal the total number of blocks transferred for both channels; thus in stereo mode, if a value of $L=64$ per channel is desired, the window length in <code>IO_PROC_CFG_WIN</code> must be defined as 128.)
Output Processing	Like the input processing domain, the output processing domain is the set of samples being used by the WOLA filterbank coprocessor to perform synthesis.

The IOP supplies a pointer to the first address that needs computation (also known as the First Address To Compute or FATC) in the `D_FATC_DATA` control register. This pointer contains the memory address of the oldest sample in the input processing domain in the input FIFO.

Details regarding the different IOP operation modes, FIFO accesses and how the FIFO accesses relate to the different IOP operation modes are described in Sections 6.2, "IOP Mono Mode " on page 51 to 6.5, "IOP Digital Mixed Mode" on page 58.

WOLA access to samples in the input and output FIFOs is handled automatically (as part of the WOLA microcode configuration) and no RCore interaction is required.

6.2 IOP MONO MODE

The mapping of the FIFOs to the Smart FIFOs for IOP mono mode is illustrated in Figure 20.

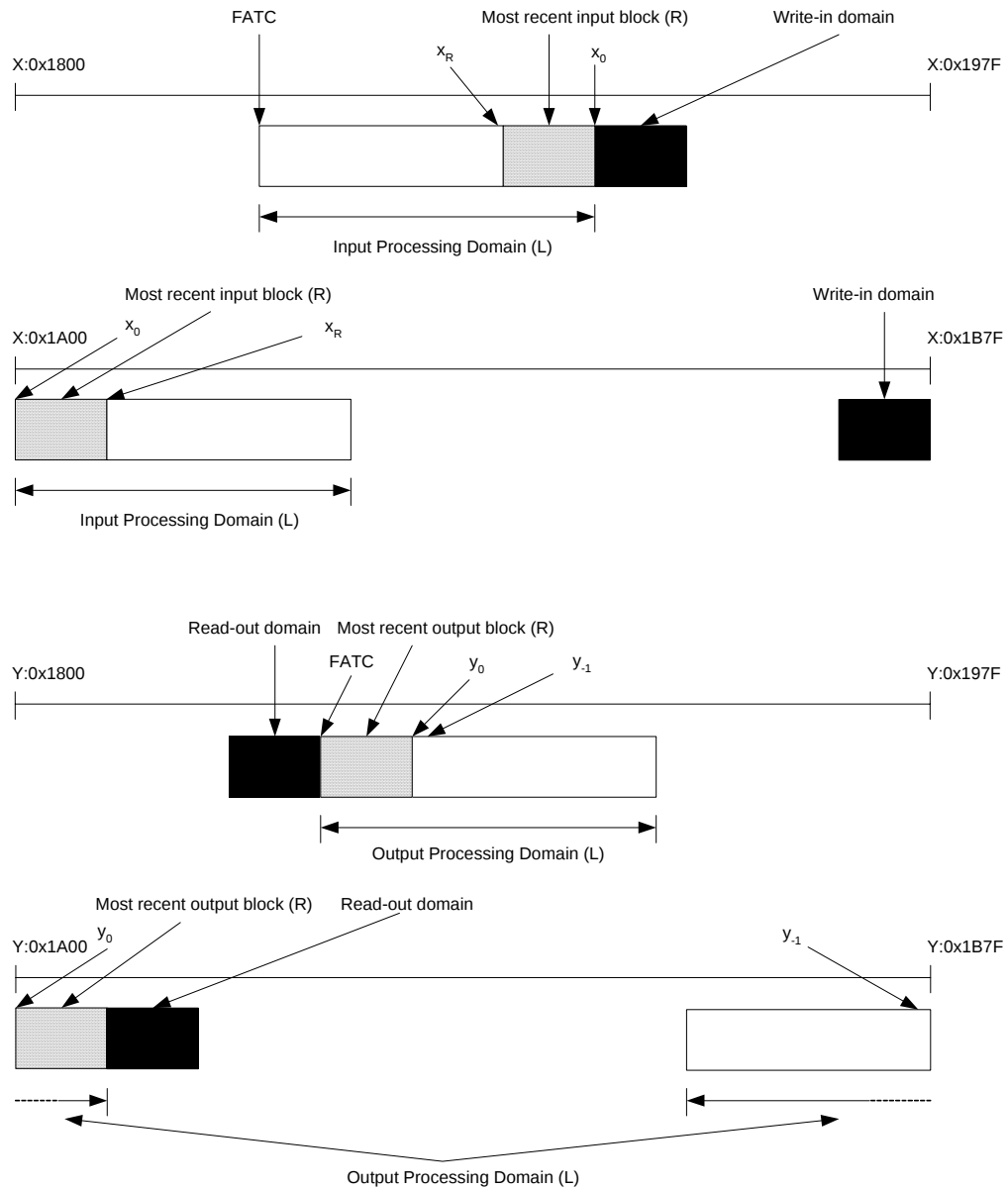


Figure 20: Smart FIFO access—mono. The upper two drawings show the normal input FIFO and the smart input FIFO. The lower two drawings show the normal output FIFO and the smart output FIFO. R is the input block size per channel and L is the analysis window length per channel.

6.2.1 IOP Mono Mode Smart Input FIFO

The smart input FIFO is created by mapping the normal input FIFO address range to another part of the X memory range in such a way that the most recent input sample, also known as x_0 , from the most recent input block of R samples, is always located at address X:0x1A00 in the smart input

FIFO. As a result the current most recent input block of R samples no longer switches location as in the normal input FIFO, but is always accessible through the same address range in the smart input FIFO.

The block of R samples that is currently being written to the input FIFO (the write-in domain) is located in the address space starting at $0x1A00 + \text{FIFO_SIZE_R}$ of the smart input FIFO. At every `IO_BLOCK_FULL` interrupt, the address mapping from the input FIFO to the smart input FIFO is updated, and all samples in the smart input FIFO (except the write-in domain) are valid and stable one `SYS_CLK` cycle after the `IO_BLOCK_FULL` interrupt has occurred. All samples are stable and accessible from the RCore until the subsequent `IO_BLOCK_FULL` interrupt occurs.

6.2.2 IOP Mono Mode Smart Output FIFO

The smart output FIFO is created by mapping the addresses from the normal output FIFO to another part of the Y-memory range in such a way that the most recent output sample from the most recent output block of R samples, also known as y_0 , is always located at address $Y:0x1A00$ in the smart output FIFO. The read-out domain samples that are being read from the output FIFO are located immediately following the most recent output block in the smart output FIFO addressing scheme.

The mapping between the output FIFO and the smart output FIFO is updated on every `IO_BLOCK_FULL` interrupt. However, because the WOLA utilizes the output FIFO memory during the synthesis process, the data in the output FIFO cannot be considered stable and valid before the WOLA has finished the synthesis process. Thus, data in the smart output FIFO may only be accessed by the RCore in the time between when the WOLA synthesis is done and the next `IO_BLOCK_FULL` interrupt occurs.

6.2.3 IOP Mono Mode Smart FIFO Address Mapping

The relation between the address $A_{r,in}$ of a sample in the normal input FIFO and the address $A_{s,in}$ of the same sample in the smart input FIFO is as follows:

$$A_{s,in} = ([(L-1) + \text{FATC} - A_{r,in}] \bmod 384) + D_SMART_IN_FIFO_BASE$$

Similarly, the relation between the address $A_{r,out}$ of a sample in the normal output FIFO and the address $A_{s,out}$ of the same sample in the smart output FIFO is as follows:

$$A_{s,out} = ([(R-1) + \text{FATC} - A_{r,out}] \bmod 384) + D_SMART_OUT_FIFO_BASE$$

6.3 IOP SIMPLE STEREO MODE

The mapping of the FIFOs to the Smart FIFOs for IOP simple stereo mode is illustrated in Figure 21.

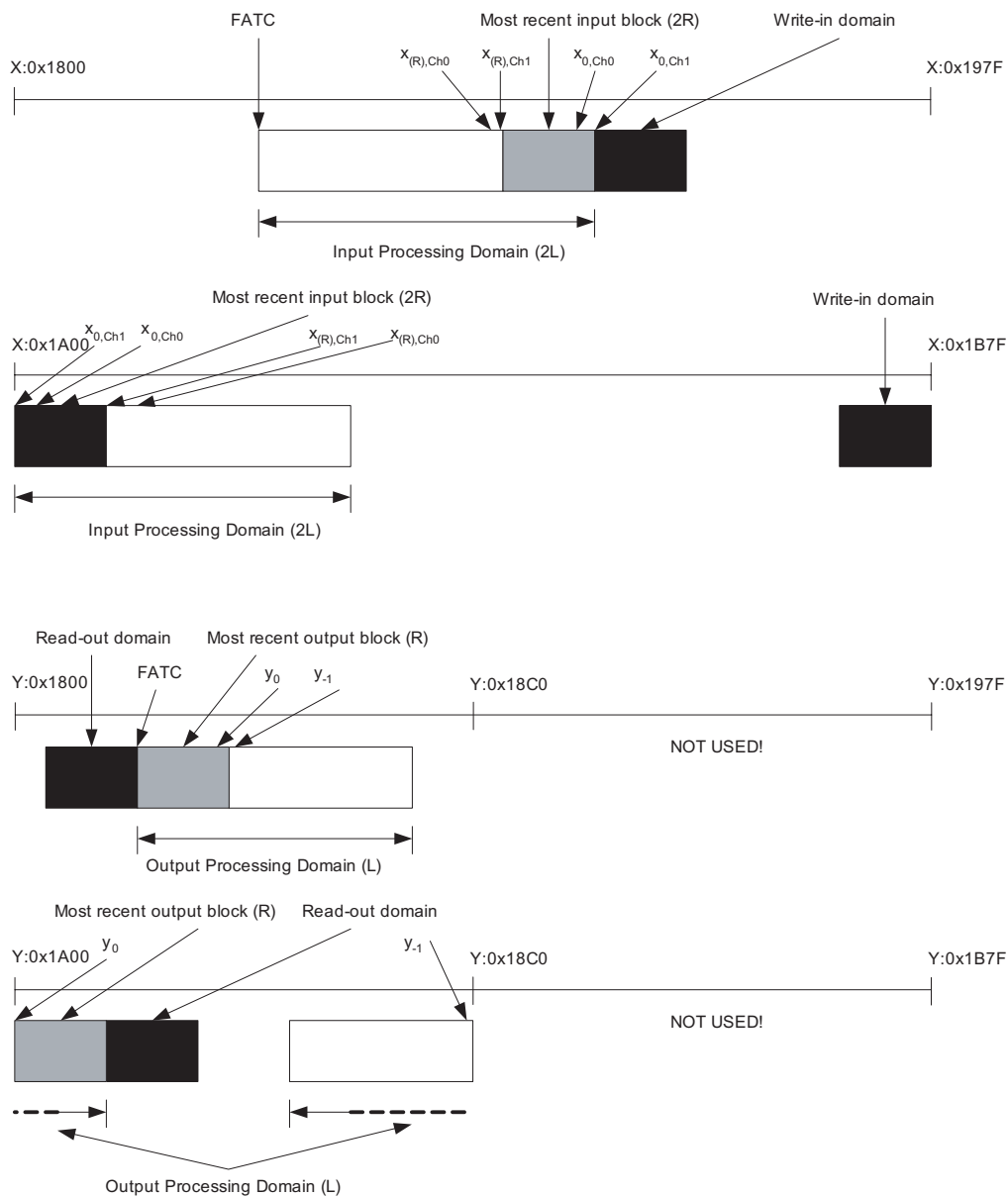


Figure 21: Smart FIFO access—simple stereo. The upper two drawings show the normal input FIFO and the smart input FIFO. The lower drawing shows the normal output FIFO and the smart output FIFO. R is the input block size per channel and L is the analysis window length per channel.

6.3.1 IOP Simple Stereo Mode Smart Input FIFO

The smart input FIFO is created by mapping the normal input FIFO address range to another part of the X memory range in such a way that the most recent input sample for channel one, also known as $x_{0,ch1}$, from the most recent input block of 2R samples is always located at address X:0x1A00 in the smart input FIFO and the most recent input sample for channel zero, also known as $x_{0,ch0}$, is placed at X:0x1A01 in the smart input FIFO.

By mapping the most recent input block of 2R samples from the input FIFO to the smart input FIFO, the most recent input block of 2R samples no longer switches location as in the normal input FIFO but is always located at the same address range in the smart input FIFO. The block of 2R samples that is currently being written to the input FIFO (the write-in domain) is located at the top of the smart input FIFO.

At each IO_BLOCK_FULL interrupt, the address mapping from the normal input FIFO to the smart input FIFO is updated and all samples in the smart input FIFO (except the write-in domain) are valid and stable one SYS_CLK cycle after the IO_BLOCK_FULL interrupt has occurred. All samples are stable and accessible from the RCore until the subsequent IO_BLOCK_FULL interrupt occurs.

6.3.2 IOP Simple Stereo Mode Smart Output FIFO

The smart output FIFO is created by mapping the addresses from the normal output FIFO to another part of the Y-memory range in such a way that the most recent output sample from the most recent output block of 2R samples, also known as y_0 , is always located at address Y:0x1A00 in the smart output FIFO. Following the most recent output block samples in the smart FIFO is the read-out domain, addressed at 2R samples after Y:0x1A00.

In this mode only the lower half of the normal and smart output FIFOs is used.

The mapping between the normal output FIFO and the smart output FIFO is updated on every IO_BLOCK_FULL interrupt. However, because the WOLA utilizes the output FIFO memory during the synthesis process, the data in the output FIFO cannot be considered stable and valid before the WOLA has finished the synthesis process. Thus, data in the smart output FIFO may only be accessed by the RCore in the time between when the WOLA is done and the next IO_BLOCK_FULL interrupt.

6.3.3 IOP Simple Stereo Mode Smart FIFO Address Mapping

The relation between the address $A_{r,in}$ of a sample in the normal input FIFO and the address $A_{s,in}$ of the same sample in the smart input FIFO is as follows:

$$A_{s,in} = ([(2L - 1) + FATC - A_{r,in}] \bmod 384) + D_SMART_IN_FIFO_BASE$$

Similarly, the relation between the address $A_{r,out}$ of a sample in the normal output FIFO and the address $A_{s,out}$ of the same sample in the smart output FIFO is as follows:

$$A_{s,out} = ([(R - 1) + FATC - A_{r,out}] \bmod 192) + D_SMART_OUT_FIFO_BASE$$

6.4 IOP FULL STEREO MODE

The mapping of the FIFOs to the Smart FIFOs for IOP full stereo mode is illustrated in Figure 22.

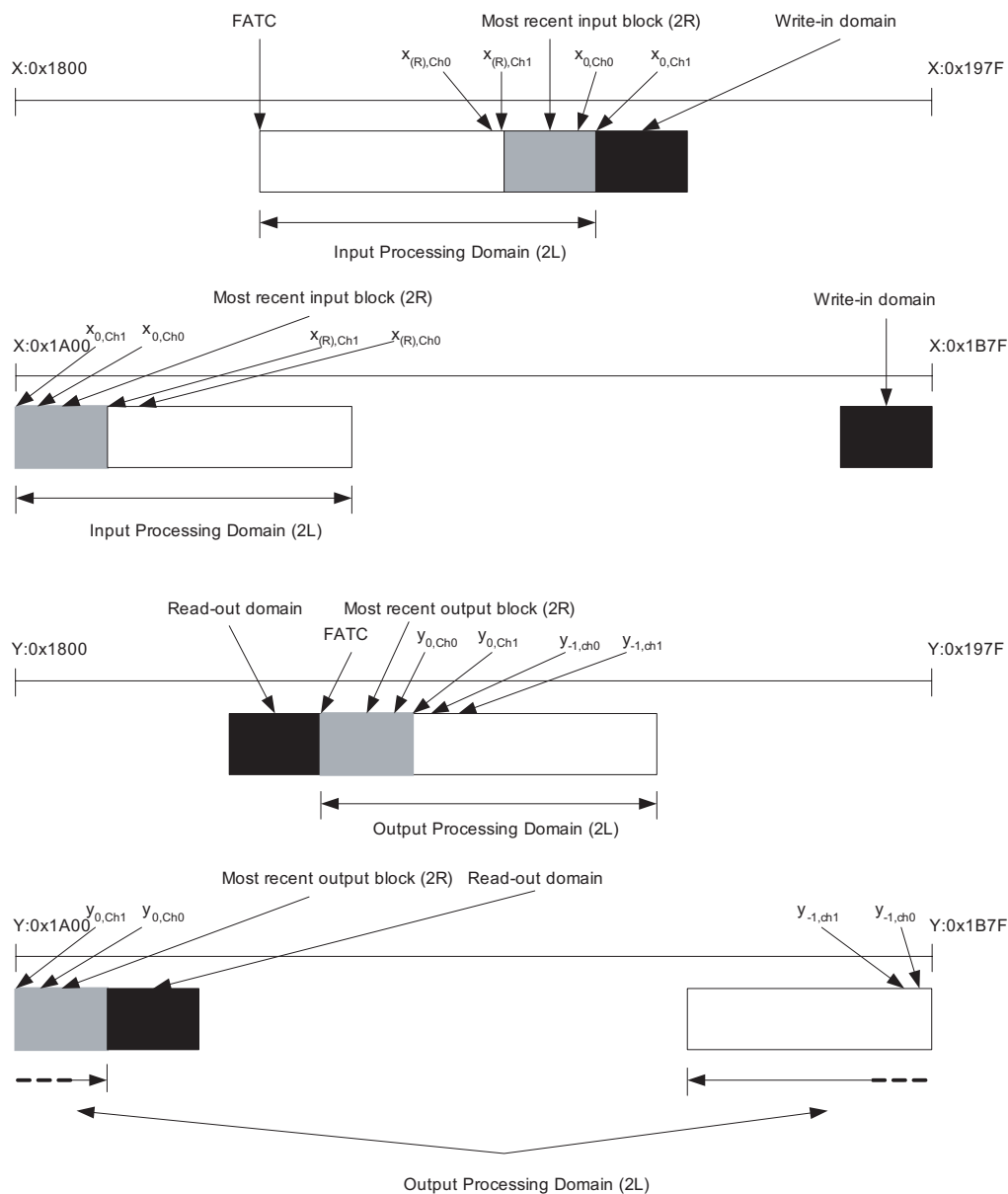


Figure 22: IOP full stereo mode. The upper two drawings show the normal input FIFO and the smart input FIFO. The lower two drawings show the normal output FIFO and the smart output FIFO. R is the input block size per channel and L is the analysis window length per channel.

6.4.1 IOP Full Stereo Mode Smart Input FIFO

The smart input FIFO is created by mapping the addresses from the normal input FIFO to another part of the X memory range in such a way that the most recent input sample from the most recent input block of R samples for channel one, also known as $x_{0,ch1}$, is always located at address $X:0x1A00$ in the smart input FIFO and the most recent sample from the most recent input block of R samples for channel zero, also known as $x_{0,ch0}$, is always located at address $X:0x1A01$ in the smart input FIFO.

By performing the FIFO mapping, the most recent input block of 2R samples no longer switches location as in the normal input FIFO but is always located at the same address range in the smart input FIFO. The block of 2R samples that is currently being written to the input FIFO (the write-in domain) is located at the top of the smart input FIFO.

At every `IO_BLOCK_FULL` interrupt the address mapping from the normal input FIFO to the smart input FIFO is updated and all samples in the smart input FIFO (except the write-in domain) are valid and stable one `SYS_CLK` cycle after the `IO_BLOCK_FULL` interrupt has occurred. All samples will be stable and accessible from the RCore until the subsequent `IO_BLOCK_FULL` interrupt occurs.

6.4.2 IOP Full Stereo Mode Smart Output FIFO

The smart output FIFO is created by mapping the addresses from the normal output FIFO to another part of the Y-memory range in such a way that the most recent output sample from the most recent output block of R samples for channel one, also known as $y_{0,ch1}$, is always located at address $Y:0x1A00$ in the smart output FIFO and the most recent sample from the most recent output block of R samples for channel zero, also known as $y_{0,ch0}$, is always located at $Y:0x1A01$. The read-out domain is addressed at $Y:0x1A00 + 2R$, following the most recent output block.

The mapping between the normal output FIFO and the smart output FIFO is updated on every `IO_BLOCK_FULL` interrupt. However, because the WOLA utilizes the output FIFO memory during the synthesis process the data in the output FIFO cannot be considered stable and valid before the WOLA has finished the synthesis process. Thus, data in the smart output FIFO may only be accessed by the RCore in the time between when the WOLA synthesis is done and the next `IO_BLOCK_FULL` interrupt.

6.4.3 IOP Full Stereo Mode Smart FIFO Address Mapping

The relation between the address $A_{r,in}$ of a sample in the normal input FIFO and the address $A_{s,in}$ of the same sample in the smart input FIFO is as follows:

$$A_{s,in} = \left(\left[(2L - 1) + FATC - A_{r,in} \right] \bmod 384 \right) + D_SMART_IN_FIFO_BASE$$

Similarly, the relation between the address location $A_{r,out}$ of a sample in the normal output FIFO and the address $A_{s,out}$ of the same sample in the smart output FIFO is as follows:

$$A_{s,out} = \left(\left[(2R - 1) + FATC - A_{r,out} \right] \bmod 384 \right) + D_SMART_OUT_FIFO_BASE$$

6.5 IOP DIGITAL MIXED MODE

The mapping of the FIFOs to the Smart FIFOs for IOP digital mixed mode is illustrated in Figure 23.

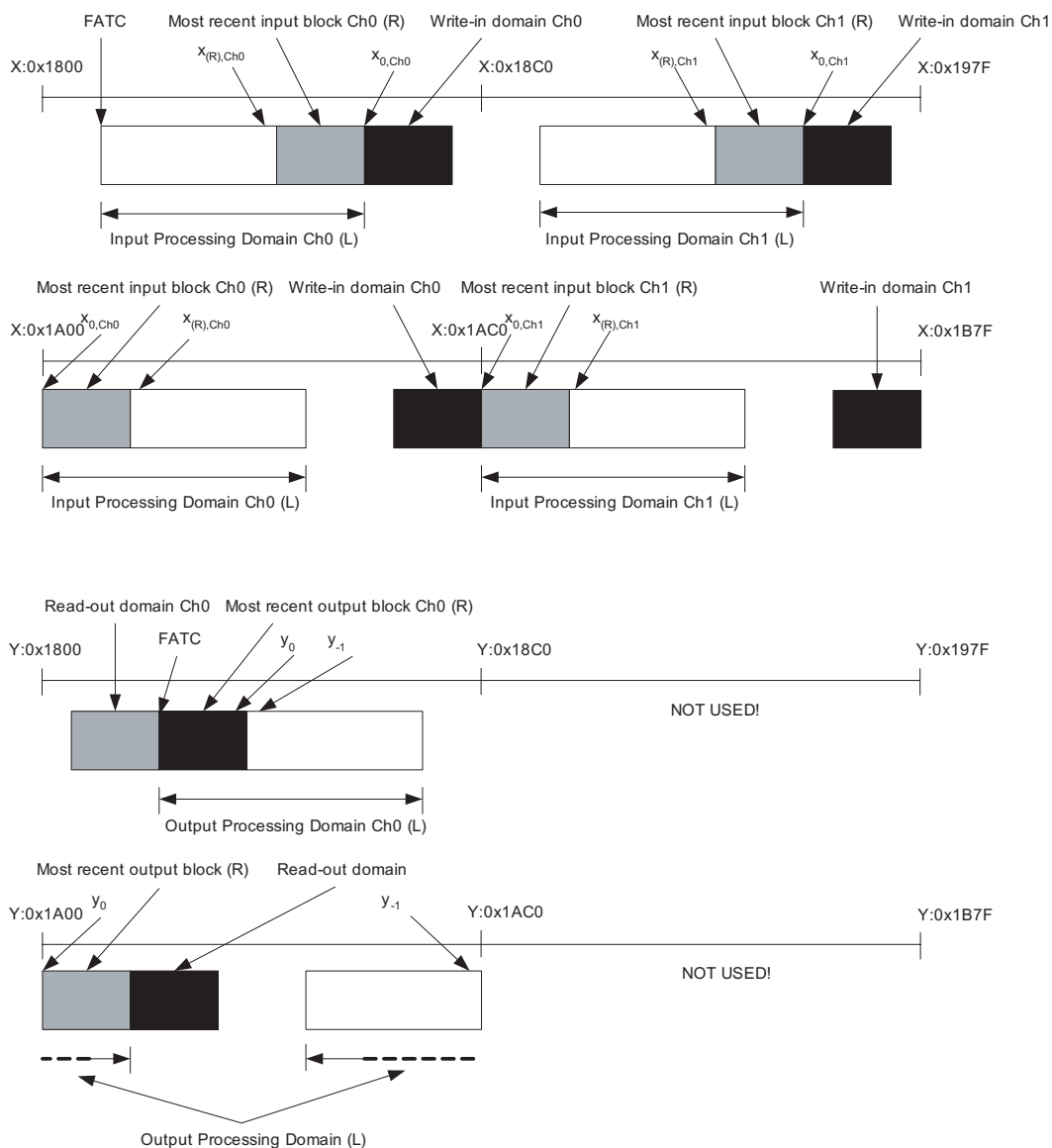


Figure 23: Smart FIFO access—digital mixed mode. The upper two drawings show the normal input FIFO and the smart input FIFO. The lower two drawings show the normal output FIFO and the smart output FIFO. R is the input block size per channel and L is the analysis window length per channel.

6.5.1 IOP Digital Mixed Mode Smart Input FIFO

The smart input FIFO is created by mapping the addresses from the normal input FIFO to another part of the X memory range in such a way that the most recent input sample from the most recent input block of R samples for channel zero, also known as $x_{0, \text{ch}0}$, is always located at address $X:0x1A00$ in the smart input FIFO and the most recent sample from the most recent input block of R samples for channel one, also known as $x_{0, \text{ch}1}$, is always located at address $X:0x1AC0$ in the smart input FIFO.

By performing the FIFO mapping, the most recent input block of R samples for each channel no longer switches location as in the normal input FIFO but is always located at the same address range in the smart input FIFO. The block of R samples for each channel that is currently being written to the input FIFO (the write-in domain) is located at the top of each segment of the smart input FIFO.

At every `IO_BLOCK_FULL` interrupt the address mapping from the normal input FIFO to the smart input FIFO is updated and all samples in the smart input FIFO, including the most recent input block of R samples for each channel, is valid and stable one `SYS_CLK` cycle after the `IO_BLOCK_FULL` interrupt has occurred. All samples are stable and accessible from the RCore until the subsequent `IO_BLOCK_FULL` interrupt occurs.

6.5.2 IOP Digital Mixed Mode Smart Output FIFO

The smart output FIFO is created by mapping the addresses from the normal output FIFO to another part of the Y-memory range in such a way that the most recent output sample from the most recent output block of R samples, also known as y_0 , is always located at address $Y:0x1A00$ in the smart output FIFO. The read-out domain for output FIFO follows the most recent sample block, at address $Y:0x1A00 + R$.

In this mode only the lower half of the normal and smart output FIFOs is used.

The mapping between the output FIFO and the smart output FIFO is updated on every `IO_BLOCK_FULL` interrupt. However, because the WOLA utilizes the output FIFO Memory during the synthesis process the data in the smart output FIFO (and therefore also in the output FIFO) cannot be considered stable and valid before the WOLA has finished the synthesis process. Thus, data in the smart output FIFO may only be accessed by the RCore in the time between when the WOLA synthesis is done and the next `IO_BLOCK_FULL` interrupt.

6.5.3 IOP Digital Mixed Mode Smart FIFO Address Mapping

The relation between the address $A_{r, \text{in}}$ of a sample in the input FIFO and the address $A_{s, \text{in}}$ of the same sample in the smart input FIFO is as follows:

Channel zero:

$$A_{s, \text{in}} = \left(\left[(L - 1) + \text{FATC} - A_{r, \text{in}} \right] \bmod 192 \right) + D_SMART_IN_FIFO_BASE$$

Channel one:

$$A_{s,in} = ([(L-1) + FATC - A_{r,in}] \bmod 192) + D_SMART_IN_FIFO_BASE + 192$$

Similarly, the relation between the address $A_{r,out}$ of a sample in the normal output FIFO and the address $A_{s,out}$ of the same sample in the smart output FIFO is as follows:

$$A_{s,out} = ([(R-1) + FATC - A_{r,out}] \bmod 192) + D_SMART_OUT_FIFO_BASE$$

6.6 IOP AUTO-MUTING

An audio muting feature (called IOP auto-muting) is implemented in the IOP. This feature is used to mute the digital output from the IOP (the IOP transmits zeros to the output stage). Some situations where this might occur are when the battery voltage is so low that only the IOP (but not the RCore or WOLA) is running properly, or where the output FIFO is not properly updated.

The auto-mute feature is enabled/disabled in the D_IO_PROC_CFG control register using the IO_PROC_CFG_AUTOMUTE bit field. Auto-muting is disabled by default.

The operation of the auto-mute feature couples tightly to the interaction between the RCore/ WOLA and the output FIFO. The time-period between two consecutive IO_BLOCK_FULL interrupts is defined as the block-tick period. If the RCore or the WOLA does not perform at least one write operation to the output FIFO during a block-tick period, the IOP will start muting the output signal at the start of the following block-tick period. This is illustrated in Figure 24.

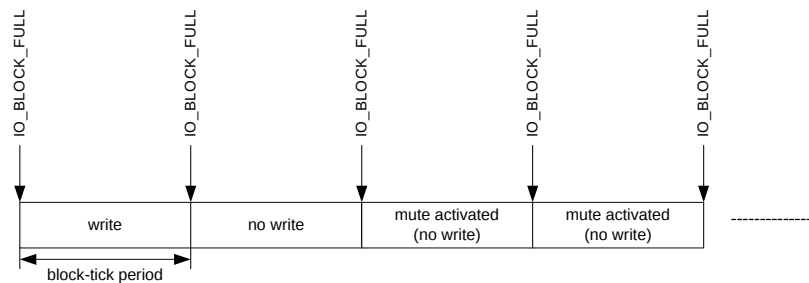


Figure 24: IOP auto-muting. The mute is activated in the block-tick period following the block-tick-period where no write operation to the output FIFO was performed.

If the RCore/WOLA starts writing to the output FIFO again, the IOP will un-mute the audio output at the start of the block-tick period following the block-tick period where the RCore/WOLA started writing to the output FIFO.

When the IOP mutes the output, the SYS_CTRL_BLK_PROCESSED bit in the D_SYS_CTRL control register is set to zero; if no muting occurs (normal operation), the SYS_CTRL_BLK_PROCESSED bit is set to one.

6.7 IOP ENABLE/DISABLE AND RESET

The IOP can be enabled or disabled in the `IO_PROC_CFG` control register. When the IOP is enabled, samples from the input stage are stored in the Input FIFO and processed samples in the Output FIFO are transmitted to the output stage. Disabling the IOP will result in the Input FIFO not being updated with new samples coming from the input stage (sampling of the input signal is maintained during an IOP-disabled condition). Similarly, output samples being stored in the output FIFO by either the WOLA filterbank or the RCore will not be transmitted to the output stage; therefore the output will be muted.

Further, it is possible to reset the IOP by writing a one to the `SYS_CTRL_IOP_RESET` bit of the `D_SYS_CTRL` register. This restores all of the internal IOP state to its default values.

6.8 CONTROL AND CONFIGURATION REGISTERS

Register Name	Register Description	Address
<code>D_SYS_CTRL</code>	System Control Register used to reset the IOP	EXT7 (RCore register)
<code>D_FATC_DATA</code>	Pointer to the First Address To Compute	Y:0x4007
<code>D_IO_PROC_CFG</code>	IOP Configuration	Y:0x8000

6.8.1 D_IO_PROC_CFG Settings

Bit Field	Field Name	Field Description
13	<code>IO_PROC_CFG_AUTOMUTE</code>	Enable/disable auto mute feature
12	<code>IO_PROC_CFG_ENABLE</code>	Enable/disable the IOP
7:6	<code>IO_PROC_CFG_WIN</code>	Analysis window length
5:3	<code>IO_PROC_CFG_BLK</code>	Input block size
2	<code>IO_PROC_CFG_CHAN_SEP</code>	Configure FIFO data organization
1	<code>IO_PROC_CFG_OUTPUT_STEREO</code>	Configure output FIFO mode
0	<code>IO_PROC_CFG_INPUT_STEREO</code>	Configure input FIFO mode

Field Name	Value Symbol	Value Description	Hex Value
IO_PROC_CFG_AUTOMUTE	IOP_AUTOMUTE_DISABLE*	Disable auto mute	0x0
	IOP_AUTOMUTE_ENABLE	Enable auto mute	0x1
IO_PROC_CFG_ENABLE	IOP_DISABLE*	Disable IOP	0x0
	IOP_ENABLE	Enable IOP	0x1
IO_PROC_CFG_WIN	IOP_WIN_32*	Window length of 32	0x0
	IOP_WIN_64	Window length of 64	0x1
	IOP_WIN_128	Window length of 128	0x2
	IOP_WIN_256	Window length of 256	0x3
IO_PROC_CFG_BLK	IOP_BLK_1	IOP block size is 1	0x0
	IOP_BLK_2	IOP block size is 2	0x1
	IOP_BLK_4	IOP block size is 4	0x2
	IOP_BLK_8*	IOP block size is 8	0x3
	IOP_BLK_16	IOP block size is 16	0x4
	IOP_BLK_32	IOP block size is 32	0x5
	IOP_BLK_64	IOP block size is 64	0x6
	IOP_BLK_128	IOP block size is 128	0x7
IO_PROC_CFG_CHAN_SEP	IOP_CHAN_INTERLEAVED*	Select separated FIFO	0x0
	IOP_CHAN_SEPARATED	Select interleaved FIFO	0x1
IO_PROC_CFG_OUTPUT_STEREO	IOP_OUTPUT_MONO*	Select mono output	0x0
	IOP_OUTPUT_STEREO	Select stereo output	0x1
IO_PROC_CFG_INPUT_STEREO	IOP_INPUT_MONO*	Select mono input	0x0
	IOP_INPUT_STEREO	Select stereo input	0x1

6.8.2 D_SYS_CTRL Settings

Bit Field	Field Name	Field Description
6	SYS_CTRL_IOP_RESET	Reset internal IOP state to default values

Field Name	Value Symbol	Value Description	Hex Value
SYS_CTRL_IOP_RESET	SYS_IOP_RESET	Reset internal IOP state to default values	0x1

7.1 ARCHITECTURE

Orela 4500 memory architecture is based on single- and dual-port memories. The layout of the memory architecture is illustrated in Figure 25.

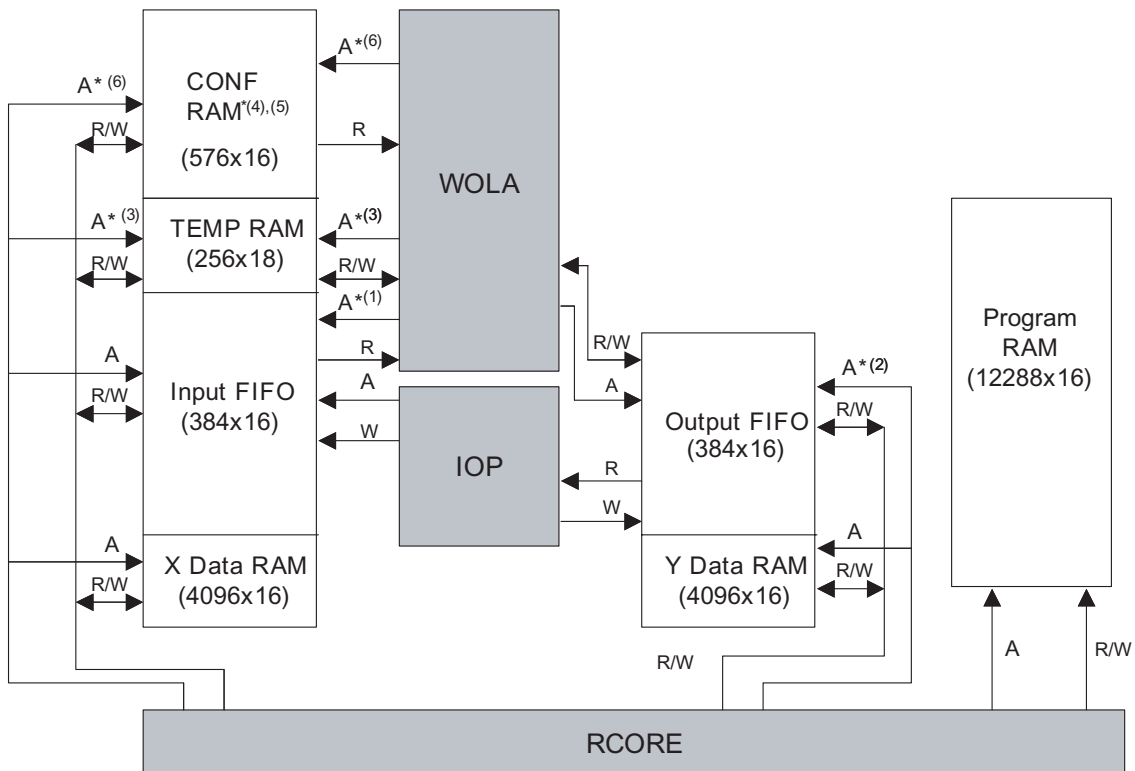


Figure 25: Orela 4500 memory architecture.

Table 8 summarizes the memory architecture.

TABLE 8: SUMMARY OF MEMORY ARCHITECTURE

Memory	Size	RCORE Access	WOLA Access	IOP Access	Memory Type
X Data	4096x16	R/W	N/A	N/A	Single-port
Y Data	4096x16	R/W	N/A	N/A	Single-port
Program	12288x16	R/W	N/A	N/A	Single-port
Input FIFO	384x16	R/W	R	W	Dual-port (1)

TABLE 8: SUMMARY OF MEMORY ARCHITECTURE (CONTINUED)

Memory	Size	RCore Access	WOLA Access	IOP Access	Memory Type
Output FIFO	384x16	R/W	R/W	R	Dual-port (2)
TEMP	256x18	R/W	R/W	N/A	Dual-port (3)
Gain (4)	256x16	R/W	R	N/A	Single-port (6)
Window (5)	Analysis: 128x16 Synthesis: 64x16	R/W	R	N/A	Single-port (6)
Microcode	128x16	R/W	R	N/A	Single-port (6)

The following points apply to the memory architecture:

1. R means read, W means write, and N/A means no access.
2. The WOLA and IOP share access to the input FIFO. The WOLA and IOP manage their own addressing. There are no restrictions on RCore access.
3. WOLA, IOP, and RCore access are shared. The WOLA needs simultaneous read and write access for the synthesis windowing. The RCore access is blocked while the WOLA is processing the synthesis windowing. If the WOLA is stopped, the RCore access is unrestricted, regardless of the IOP state.
4. The WOLA and RCore share access to this memory. While the WOLA is running, access by the RCore is blocked, because the WOLA utilizes both ports simultaneously.
5. N+1 gains are specified (where N=128 is the maximum number of channels).
6. Although the window length can be 256 for analysis and 128 for synthesis, only half the length of each window is stored because the windows are symmetric.
7. The RCore and the WOLA share access to these memories; the address lines are multiplexed. The RCore access is blocked while the WOLA is running.

7.2 PROGRAM MEMORY

The program memory has an address space from 0x0000 to 0x3FFF.

In addition to the 16-word interrupt vector table located at 0x3FF0 to 0x3FFF, and the 12 Kwords of Program RAM that are available in the memory range 0x1000 to 0x3FEF, the system also features a Program ROM, located at 0x0000 to 0x03FF.

This Program ROM contains boot code as well as system-level routines that are used for accessing EEPROMs, etc. For more information about the Program ROM see the *Orela 4500 Firmware Reference Manual*.

The interrupt vectors and their locations are shown in Table 9.

TABLE 9: INTERRUPT VECTORS

Interrupt	Vector	Address	Interrupt #
WOLA_DONE	D_VECT_WOLADONE	P:0x3FF0	0
IO_BLOCK_FULL	D_VECT_IOBLOCK	P:0x3FF1	1
PCM	D_VECT_PCM	P:0x3FF2	2

TABLE 9: INTERRUPT VECTORS (CONTINUED)

Interrupt	Vector	Address	Interrupt #
UART_RX	D_VECT_UART_RX	P:0x3FF3	3
UART_TX	D_VECT_UART_TX	P:0x3FF4	4
TIMER	D_VECT_TIMER	P:0x3FF5	5
WATCHDOG	D_VECT_WATCHDOG	P:0x3FF6	6
SPI	D_VECT_SPI	P:0x3FF7	7
TWSS	D_VECT_TWSS	P:0x3FF8	8
REMOTE	D_VECT_REMOTE	P:0x3FF9	9
EXT3_RX	D_VECT_EXT3_RX	P:0x3FFA	10
EXT3_TX	D_VECT_EXT3_TX	P:0x3FFB	11
GPIO	D_VECT_GPIO	P:0x3FFC	12

7.3 DATA MEMORY

The data memory of Orela 4500 consists of two 4-Kword single-port memories, called XRAM and YRAM. They are located at 0x0000 of X memory and 0x0000 of Y memory respectively. At higher address locations in X and Y memory, other memory blocks and control registers are mapped into the address spaces for various auxiliary functions. For more information, see Sections 7.1, “Architecture” on page 63 and 7.4, “Memory Maps” on page 65.

7.4 MEMORY MAPS

The organization of the memory spaces is illustrated in Figure 26. All start and end addresses of the active parts of the memory spaces are associated with `#define` statements that can be found in the `<sk25_hw.inc>` file.

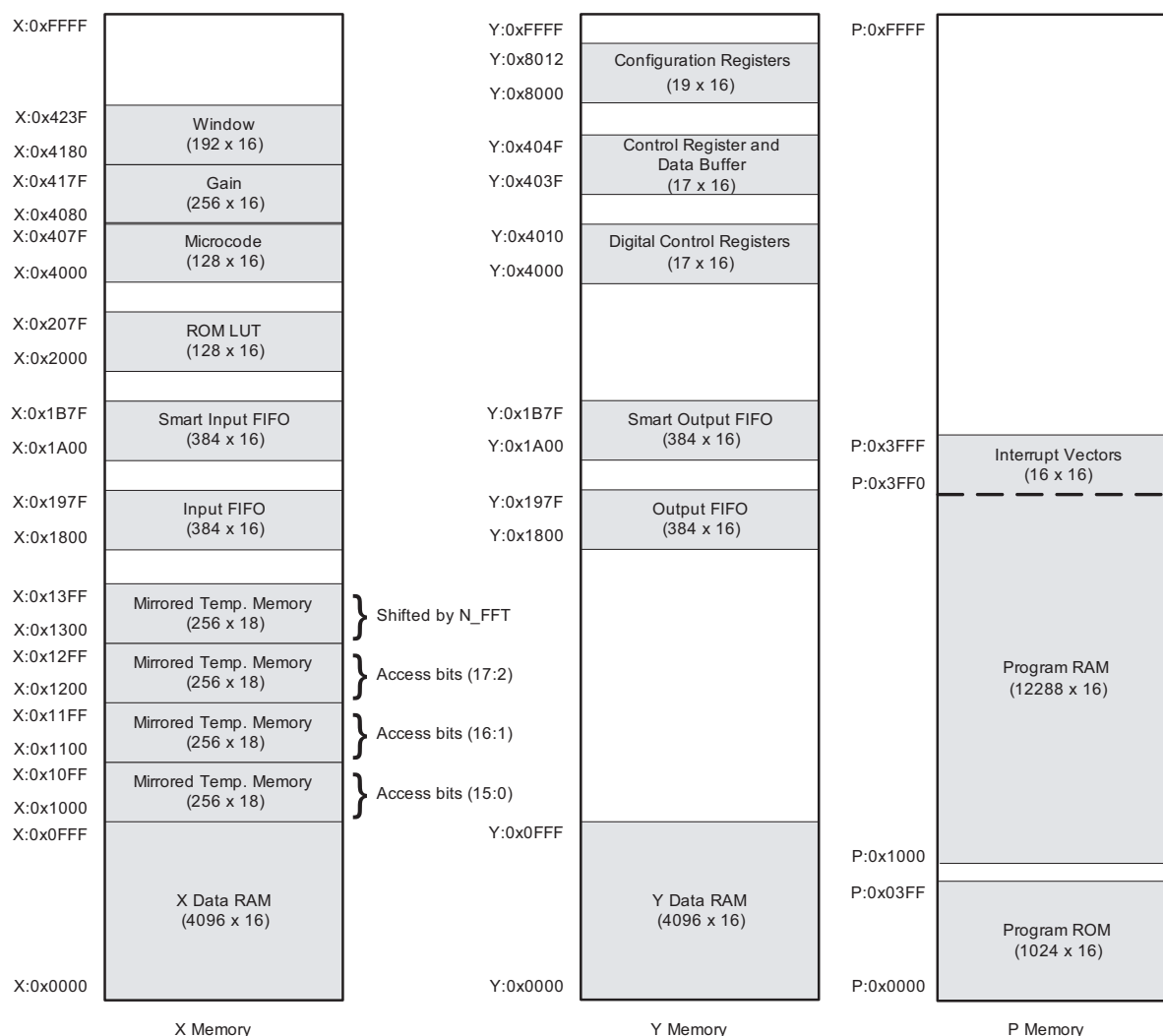


Figure 26: Orela 4500 X,Y and P memory maps.

7.5 DATA ROM

Orela 4500 features a set of look-up tables located in ROM. These look-up tables are used to calculate reciprocals, logarithms, etc. The look-up tables are described in more detail in the *Orela 4500 Firmware Reference Manual*.

7.6 PROGRAM ROM

Orela 4500 features basic operations, functions and command sets located in ROM. These elements are used to provide functional support for many of the peripherals and interfaces

available on the system. The Program ROM is described in more detail in the *Orela 4500 Firmware Reference Manual*.

7.7 MEMORY CONTROL AND CONFIGURATION REGISTERS

The Orela 4500 chip uses both memory-mapped and register-based control for configuration and operation of the Orela 4500 system peripherals. Two sections of memory are dedicated to system control and configuration, located at Y:0x4000 and Y:0x8000 respectively. In the Y:0x4000 section there is an additional memory mapped register located at Y:0x403F for controlling the PCM interface which is followed by the PCM data buffer at address Y:0x4040.

The different registers and their configuration options are referenced in the appropriate chapters of this reference manual with a summary of the register details available in Appendix C, "Control and Configuration Register Overview" on page 141.

This page left blank for printing purposes.

8.1 GENERAL PURPOSE I/O (GPIO)

Sixteen software-controlled general-purpose digital I/O (GPIO) pins are provided. The function of these GPIO pins is completely user-defined, whether as input or output, per bit or per word.

Almost all GPIO pins are multiplexed with other functionality to reduce pin-out, and the desired functionality is configured in the `D_GPIO_PIN_CFG` register using the `GPIO_PIN_CFG_<interface>` bit fields listed in the configuration information of Section 8.1.2, “Control and Configuration Registers” on page 70. For more information about the GPIO pins and other pins, see the Orela 4500 series datasheets.

When configured for GPIO functionality the direction of each individual GPIO pin can be configured in the `D_GPIO_DIR_CFG` register. Setting the direction bit to 1 results in the associated GPIO pin configured as an input. Setting the direction bit to 0 results in the pin configured as an output. The default for all GPIO pins is input.

When not configured for GPIO functionality, the direction of the pins must be set appropriately for the other functionality. For example, if `UART_TX` functionality is desired on a multiplexed pin, this pin must be configured as an output in the `D_GPIO_DIR_CFG` control register.

GPIO pins configured as inputs have an internal pull-up resistor. Thus, a read from GPIO pin configured as an input that is left floating will result in a data value of 1.

Writing to pins configured as inputs through `D_GPIO_DATA` is permitted and has no effect on these pins. A read from pins through `D_GPIO_DATA` always returns the value at the physical pin, not the data stored in the output register, regardless of the configured direction for that pin.

Detailed descriptions of how to operate interfaces when the pins are not configured for GPIO functionality can be found in each interface's section of this manual.

The input/output data of the GPIO interface is accessed through the `D_GPIO_DATA` register.

8.1.1 Interrupt

GPIO inputs have the option of being associated with a configurable interrupt. See Section 9.1, “Interrupt Controller” on page 101 for more information about interrupt configuration and handling.

The GPIO interrupt is configured using the `D_GPIO_INT_CFG` register and may be configured to trigger on a variety of events from up to two of the 16 GPIO input pins. These two sources are referred to as source A and source B, which may be selected using the `GPIO_INT_CFG_SRC_A` and `GPIO_INT_CFG_SRC_B` bit fields of the `D_GPIO_INT_CFG` register respectively. The interrupt as seen by the interrupt controller is generated by performing a logical OR on the interrupts generated by sources A and B.

Each source may be configured to generate an interrupt on one type of event. The `GPIO_INT_CFG_EVENT_A` and `GPIO_INT_CFG_EVENT_B` bit fields from `D_GPIO_INT_CFG` may respectively be used to select the corresponding interrupt triggering event for source A and source B. The possible triggering events are listed in Table 10.

TABLE 10: GPIO INTERRUPT EVENTS

Event	Event description
Disabled	Interrupt is never generated
High Level	Interrupt is generated whenever a logical 1 is detected on the GPIO input pin. Interrupts continue to be generated while this condition is true.
Low Level	Interrupt is generated whenever a logical 0 is detected on the GPIO input pin. Interrupts continue to be generated while this condition is true.
Rising Edge	Interrupt is generated when a transition from a logical 0 to a logical 1 is detected.
Falling Edge	Interrupt is generated when a transition from a logical 1 to a logical 0 is detected.
Transition	Interrupt is generated when a transition from a logical 0 to a logical 1 or a transition from a logical 1 to a logical 0 is detected.

For convenience, the macro `Set_GPIO_Int_Config` has been provided in the BOS include file (*boss.inc*) to be used in configuring the triggering sources and events for the GPIO interrupt. For more information about configuring the GPIO interrupt using `Set_GPIO_Int_Config` see the *Orela 4500 Firmware Reference Manual*.

8.1.2 Control and Configuration Registers

Register Name	Register Description	Address
<code>D_GPIO_DATA</code>	GPIO data	Y:0x4006
<code>D_GPIO_DIR_CFG</code>	GPIO pin I/O direction configuration	Y:0x8006
<code>D_GPIO_PIN_CFG</code>	GPIO pin multiplex configuration	Y:0x8007
<code>D_GPIO_INT_CFG</code>	GPIO interrupt configuration	Y:0x8012

8.1.2.1 D_GPIO_PIN_CFG Settings

Bit Field	Field Name	Field Descriptions
15	GPIO_PIN_CFG_I2S	Enable/disable the I ² S interface
14	GPIO_PIN_CFG_REMOTE	Enable/disable the use of wireless receiver remote control interface
12	GPIO_PIN_CFG_UART	Enable/disable the UART interface
11	GPIO_PIN_CFG_PCM	Enable/disable the PCM interface
10	GPIO_PIN_CFG_UCLK	Enable/disable the UCLK output pin
9	GPIO_PIN_CFG_LSAD3	Enable/disable the LSAD 3 input
8	GPIO_PIN_CFG_LSAD4	Enable/disable the LSAD 4 input
7	GPIO_PIN_CFG_LSAD5	Enable/disable the LSAD 5 input
6	GPIO_PIN_CFG_LSAD2	Enable/disable the LSAD 2 input
5	GPIO_PIN_CFG_LSAD1	Enable/disable the LSAD 1 input
4	GPIO_PIN_CFG_LSAD0	Enable/disable the LSAD 0 input
3	GPIO_PIN_CFG_CLKRST_EBL	Enable/disable multichip synchronization.

Field Name	Value Symbol	Value Description	Hex Value
GPIO_PIN_CFG_I2S	GPIO_DISABLE_I2S*	Disable I ² S interface	0x0
	GPIO_ENABLE_I2S	Enable the I ² S interface	0x1
GPIO_PIN_CFG_REMOTE	GPIO_DISABLE_REMOTE*	Disable use of wireless	0x0
	GPIO_ENABLE_REMOTE	Enable use of wireless	0x1
GPIO_PIN_CFG_UART	GPIO_DISABLE_UART*	Disable the UART interface	0x0
	GPIO_ENABLE_UART	Enable the UART interface	0x1
GPIO_PIN_CFG_PCM	GPIO_DISABLE_PCM*	Disable the PCM interface	0x0
	GPIO_ENABLE_PCM	Enable the PCM interface	0x1
GPIO_PIN_CFG_UCLK	GPIO_DISABLE_UCLK*	Disable the UCLK pin	0x0
	GPIO_ENABLE_UCLK	Enable the UCLK pin	0x1
GPIO_PIN_CFG_LSAD5	GPIO_DISABLE_LSAD5*	Disable input LSAD 5	0x0
	GPIO_ENABLE_LSAD5	Enable input LSAD 5	0x1
GPIO_PIN_CFG_LSAD4	GPIO_DISABLE_LSAD4*	Disable input LSAD 4	0x0
	GPIO_ENABLE_LSAD4	Enable input LSAD 4	0x1
GPIO_PIN_CFG_LSAD3	GPIO_DISABLE_LSAD3*	Disable input LSAD 3	0x0
	GPIO_ENABLE_LSAD3	Enable input LSAD 3	0x1
GPIO_PIN_CFG_LSAD2	GPIO_DISABLE_LSAD2*	Disable input LSAD 2	0x0
	GPIO_ENABLE_LSAD2	Enable input LSAD 2	0x1
GPIO_PIN_CFG_LSAD1	GPIO_DISABLE_LSAD1*	Disable input LSAD 1	0x0
	GPIO_ENABLE_LSAD1	Enable input LSAD 1	0x1
GPIO_PIN_CFG_LSAD0	GPIO_DISABLE_LSAD0*	Disable input LSAD 0	0x0
	GPIO_ENABLE_LSAD0	Enable input LSAD 0	0x1
GPIO_PIN_CFG_CLKRST_EBL	GPIO_DISABLE_CLKDIV_RESET*	Disable multi-chip sync.	0x0
	GPIO_ENABLE_CLKDIV_RESET	Enable multi-chip sync.	0x1

8.1.2.2 D_GPIO_INT_CFG Settings

Bit Field	Field Name	Field Description
13:10	GPIO_INT_CFG_SRC_A	Selection of source A for the GPIO interrupt
9:7	GPIO_INT_CFG_EVENT_A	Selection of the source A triggering event for the GPIO interrupt
6:3	GPIO_INT_CFG_SRC_B	Selection of source B for the GPIO interrupt
2:0	GPIO_INT_CFG_EVENT_B	Selection of the source B triggering event for the GPIO interrupt

Field Name	Value Symbol	Value Description	Hex Value
GPIO_INT_CFG_EVENT_A	GPIO_INT_EVENT_DISABLED*	Disable GPIO interrupt source	0x0
	GPIO_INT_EVENT_HIGH_LEVEL	Interrupt trigger on logical 1	0x1
	GPIO_INT_EVENT_LOW_LEVEL	Interrupt trigger on logical 0	0x2
	GPIO_INT_EVENT_RISING_EDGE	Interrupt trigger on rising edge	0x3
	GPIO_INT_EVENT_FALLING_EDGE	Interrupt trigger on falling edge	0x4
	GPIO_INT_EVENT_TRANSITION	Interrupt trigger on either edge	0x5
	N/A	Disable GPIO interrupt source	0x6
	N/A	Disable GPIO interrupt source	0x7
GPIO_INT_CFG_EVENT_B	GPIO_INT_EVENT_DISABLED*	Disable GPIO interrupt source	0x0
	GPIO_INT_EVENT_HIGH_LEVEL	Interrupt trigger on logical 1	0x1
	GPIO_INT_EVENT_LOW_LEVEL	Interrupt trigger on logical 0	0x2
	GPIO_INT_EVENT_RISING_EDGE	Interrupt trigger on rising edge	0x3
	GPIO_INT_EVENT_FALLING_EDGE	Interrupt trigger on falling edge	0x4
	GPIO_INT_EVENT_TRANSITION	Interrupt trigger on either edge	0x5
	N/A	Disable GPIO interrupt source	0x6
	N/A	Disable GPIO interrupt source	0x7

8.2 UART

The general-purpose UART is designed to support a standard RS-232 transmission protocol. The UART uses a standard data format with one start bit, eight data bits and one stop bit. Standard communication speeds (9600 to 115200 bps) are supported.

The UART receive (UART_RX) and transmit (UART_TX) lines are multiplexed with other functionality. See Section 8.1, “General Purpose I/O (GPIO)” on page 69 for more information about how to configure the pins for UART functionality.

The UART operation is configured with the UART_CFG_ENABLE and UART_CFG_PRESCALE bit fields in the D_UART_CFG configuration register. The UART_CFG_PRESCALE bit in the D_UART_CFG register determines whether a divide-by-12 clock prescaler is used.

The speed of the UART is determined by the 16-bit value of the D_UART_SPEED_CFG register. The baud rate (specified in bits/sec) may be calculated using this value and the PCLK rate according to the following equation. For a baud rate of 115200 bits/sec, set D_UART_SPEED_CFG to 23593 for a PCLK frequency of 1.28 MHz.

$$R_{BAUD} = 6,250,000 \cdot \left(\frac{D_UART_SPEED_CFG}{PCLK} \right)$$

Data transmission and reception is done via the D_UART_DATA register.

For more information about PCLK configuration, see Section 9.3, “Oscillator Circuitry and Clock Generation” on page 107.

8.2.1 Interrupts

The UART port has interrupts associated with the transmission and reception of values to/from the D_UART_DATA register. See Section 9.1, “Interrupt Controller” on page 101 for information regarding interrupt configuration and handling.

The UART_TX interrupt is generated when the value in the D_UART_DATA register has been written to the (non-visible) transmit shift register and a new value can be loaded into the UART_DATA register. The UART_RX interrupt is generated after a value has been successfully received in the (non-visible) receive shift register and has been written to the D_UART_DATA register.

8.2.2 Control and Configuration Registers

Register Name	Register Description	Address
D_UART_DATA	UART data Rx/Tx register (bits 7:0)	Y:0x4008
D_UART_CFG	UART configuration	Y:0x8003
D_UART_SPEED_CFG	UART speed	Y:0x8004

8.2.2.1 D_UART_CFG Settings

Bit Field	Field Name	Field Description
1	UART_CFG_ENABLE	Enable/disable UART
0	UART_CFG_PRESCALE	Enable/disable prescaler

Field Name	Value Symbol	Value Description	Hex Value
UART_CFG_ENABLE	UART_DISABLE*	Disable UART	0x0
	UART_ENABLE	Enable UART	0x1
UART_CFG_PRESCALE	UART_FAST*	Do not prescale clock	0x0
	UART_SLOW	Prescale clock by 12	0x1

8.3 I²S INTERFACE

The I²S interface provides Philips I²S compatible serial interfaces to the inputs and outputs of both the analog and digital portions of the Orela 4500 chip. This allows for the direct interfacing of the Orela 4500 chip with other audio components that have a similar interface, as well as synchronizing the data and data samples being processed in applications based on multi-chip Orela 4500 configurations. A diagram of the I²S interface and its connections can be found in Figure 27.

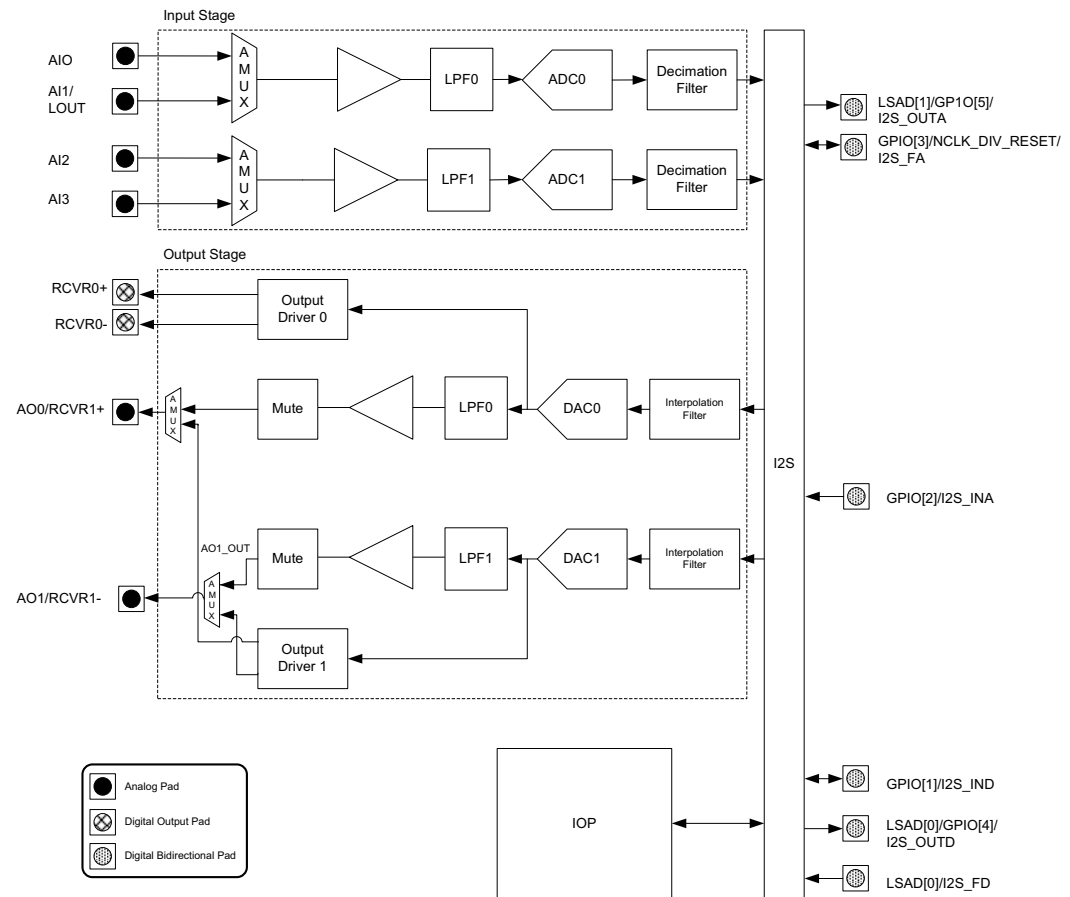


Figure 27: I²S Interface layout and connections.

Note: All I2S_* pins are multiplexed with GPIO functionality.

The I²S interface serial connections are multiplexed with GPIO pins that must be configured appropriately as inputs or outputs using the `D_GPIO_DIR_CFG` register. A complete list and description of the interface connections along with the direction options for each connection is available in Table 11 below. Further, the overall interface may be enabled or disabled using `GPIO_PIN_CFG_I2S` bit from the `D_GPIO_PIN_CFG` register. Configuration of both of these registers is outlined in Section 8.1, “General Purpose I/O (GPIO)” on page 69. There are no I²S interface specific configuration registers.

TABLE 11: I²S INTERFACE CONNECTIONS

Connection	Direction	Description
PCLK (Internal connection)	N/A	Clock for controlling data transmission. For more information regarding configuration of PCLK see Section 10.3.
I2S_FD	Input	Digital frame signal. Used for selecting which of the two channels is being transmitted or received by the digital circuitry.
I2S_IND	Input	Digital input serial data line. Used as the input signal to the IOP in place of the input stage analog circuitry.
I2S_INA	Input	Analog input serial data line. Used as the input signal to the D/A converters of the output stage in place of the output stage digital circuitry.
I2S_FA	Input/Output	Analog frame signal. Used for selecting which of the two channels is being transmitted or received by the analog circuitry.
I2S_OUTD	Output	Digital output serial data line. The output signal from the IOP after processing by the RCore DSP and WOLA.
I2S_OUTA	Output	Analog output serial data line. The output signal from A/D converters of the input stage.

The I²S interface can be configured to operate in I²S master mode or in I²S slave mode. In both modes the digital I²S connections behave as a slave, with the input frame control signal coming from I2S_FD only in master mode. This frame signal controls the input data transmission using signal I2S_IND and controls the output data transmission using signal I2S_OUTD. The analog I²S frame control signal (I2S_FA) is configured as an input in slave mode, allowing external control of data flow along I2S_INA and I2S_OUTA, or as an output in master mode, allowing the analog circuitry to control the data flow into and out of the analog circuitry along these data connections.

For convenience the macro GPIO_MUX_Select has been provided in *boss.inc*. It may be used with the options SELECT_I2S_MASTER or SELECT_I2S_SLAVE to configure the D_GPIO_DIR_CFG and D_GPIO_PIN_CFG registers for I²S operation. For more information about configuring the I²S interface using GPIO_MUX_Select see the *Orela 4500 Firmware Reference Manual*.

Each I²S connection that is part of the I²S interface uses three signals: the peripheral clock (PCLK), a frame signal and a serial data line. Audio data that is transmitted over the I²S interface is clocked by PCLK with each data bit arriving with a falling clock edge and being latched in on the following rising clock edge. The data for a channel is transmitted serially in 16-bit two's-complement notation starting with the most significant bit (MSB). Both sides of the interface must operate on the same PCLK to ensure correct data transmission/reception.

Up to two channels of data may be transmitted along each serial data line in each data frame. The frame signals, which are also synchronized with the falling edge of PCLK, are used to control the data flow along each of the serial data lines as noted above indicating the start of each 16-bit subframe. A data transfer starts one clock cycle after a transition on its associated frame signal. When the frame signal is low, the data transferred should be from channel 0. When the frame signal is high, the data transferred should be from channel 1. Figure 28 below shows the transmission of data across one-half of the I²S interface.

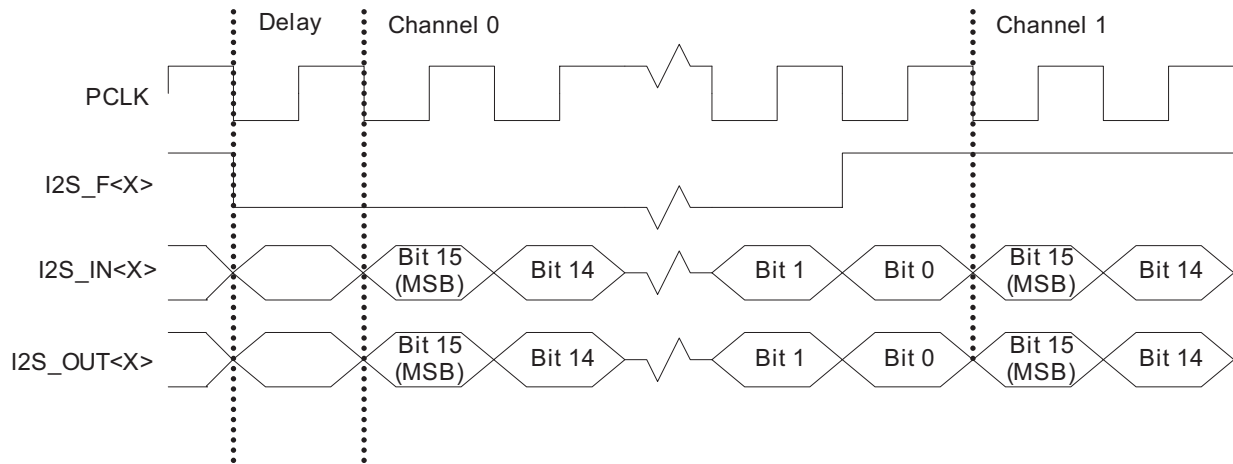


Figure 28: I²S interface data transmission operation

Since the transmission from a transmitter to a receiver is clocked, with the serial data and frame signal lines synchronized to the falling edge of the PCLK, it is important for external controllers to synchronize input signals with the PCLK through the UCLK interface. For more information regarding the configuration of UCLK see Section 9.3, “Oscillator Circuitry and Clock Generation” on page 107.

8.4 LOW-SPEED A/D CONVERTERS

A total of six external low-speed 10-bit A/D (LSAD) converter inputs are available on Orela 4500. In addition to the external input lines, the LSAD circuitry also reads the power supply voltage as an internally hardwired input. Furthermore, one channel is used for internal purposes (monitoring the VREG line), which brings the total number of LSAD converter inputs to eight. Converted data originating from the internal channel associated with the VREG line cannot be read from the RCore.

The purpose of the LSAD converter is to sample input voltages that may change slowly—for example, the voltage associated with a potentiometer-based volume control. All LSAD converter inputs are multiplexed with other functionality. Refer to Section 8.1, “General Purpose I/O (GPIO)” on page 69 for information on how to configure these multiplexed pins for LSAD converter functionality.

The LSAD inputs are sampled sequentially at a sampling frequency determined by the frequency of MCLK and prescaled by the value of the DIV_CLK_CTRL_LSAD_FREQ_CTRL bit field in the A_DIV_CLK_CTRL control register.

The relationship between the sampling frequency, the frequency of MCLK, and the setting of the DIV_CLK_CTRL_LSAD_FREQ_CTRL bit field in the A_DIV_CLK_CTRL control register is shown below:

$$f_{LSAD} = \frac{f_{MCLK}}{20 \cdot (DIV_CLK_CTRL_LSAD_FREQ_CTRL + 1)}$$

At a 1.28 MHz MCLK frequency the default sampling frequency is 12.8 kHz, which gives an effective sampling frequency of 1.6 kHz for each of the eight LSAD channels.

The converted value of a specific input channel is stored in two analog control registers:

- The `A_LSAD_<channel>_DATA` control register contains the eight most significant bits (MSBs).
- The `LSAD_CTRL_LSB` bit field of the `A_LSAD_<channel>_CTRL` control register contains the two least significant bits (LSBs) of the converted value. This control register also contains information about how to configure the specific input channel.

In case more than the eight MSBs are desired, the two LSBs must be read immediately following the reading of the eight MSBs; otherwise the LSBs that are read are not guaranteed to associate with the MSBs that have been read. (See the sample code for an example of how to achieve this.)

To accommodate a wide range of transducers on the LSAD inputs, the dynamic range of each individual input can be configured to a specified range. The native dynamic range of the LSAD is a voltage range going from -1 V to 3 V, where the converted value is represented by a 10-bit two's complement number. As such, an input value of -1 V will convert into the value -512 and the input value 3 V will convert into the value 511 when using the native (default) input dynamic range.

For all other selectable input dynamic ranges, the converted value is represented by an 8-bit unsigned value; only the eight MSBs in the `A_LSAD_<channel>_DATA` register are valid. This value will be 0 (hex value 0x00) when the LSAD input is at or below the low end of the selected range, and 255 (hex value 0xFF) when the input is at or above the high end of the selected range.

The desired input dynamic range of the six LSAD converter inputs can be configured with the `LSAD_CTRL_RANGE` bits in the `A_LSAD_<channel>_CTRL` registers. All available input dynamic ranges can be obtained from the configuration information Section 8.4.1, "Control and Configuration Registers".

For battery monitoring, the value of the power supply voltage applied on VBAT can also be converted. The converted value of the power supply voltage is stored in the `A_LSAD_VBAT_DATA` (eight MSBs) control register and the `A_LSAD_VBAT_CTRL` (two LSBs) control register. The dynamic range selection for the internal VBAT LSAD input is exactly the same as for the six external LSAD inputs.

Note: Due to the protection diodes in the LSAD input pads that start conducting at an input voltage below -0.3 V and at an input voltage above 1.5 V (in low-voltage mode), converted values that represent input voltages below and above these voltages will be incorrect. Input voltages below and above these two voltages should be avoided.

8.4.1 Control and Configuration Registers

Register Name	Register Description	Address
A_LSAD_CH0_DATA	LSAD channel 0 data	A:0x00
A_LSAD_CH0_CTRL	LSAD channel 0 control	A:0x01
A_LSAD_CH1_DATA	LSAD channel 1 data	A:0x02
A_LSAD_CH1_CTRL	LSAD channel 1 control	A:0x03
A_LSAD_CH2_DATA	LSAD channel 2 data	A:0x04
A_LSAD_CH2_CTRL	LSAD channel 2 control	A:0x05
A_LSAD_CH3_DATA	LSAD channel 3 data	A:0x06
A_LSAD_CH3_CTRL	LSAD channel 3 control	A:0x07
A_LSAD_CH4_DATA	LSAD channel 4 data	A:0x08
A_LSAD_CH4_CTRL	LSAD channel 4 control	A:0x09
A_LSAD_CH5_DATA	LSAD channel 5 data	A:0x0A
A_LSAD_CH5_CTRL	LSAD channel 5 control	A:0x0B
A_LSAD_VBAT_DATA	Supply voltage (VBAT) data	A:0x0C
A_LSAD_VBAT_CTRL	VBAT sampling control	A:0x0D
A_DIV_CLK_CTRL	LSAD sampling frequency control	A:0x13

8.4.1.1 A_LSAD_<channel>_CTRL Settings

Bit Field	Field Name	Field Description
4:2	LSAD_CTRL_RANGE	LSAD input dynamic range settings
1:0	LSAD_CTRL_LSB	LSBs of 10-bit converted value

Field Name	Value Symbol	Value Description	Hex Value
LSAD_CTRL_RANGE	LSAD_RANGE_N1V_3V*	Dynamic range is -1 V to 3 V	0x0
	LSAD_RANGE_1V_3V	Dynamic range is 1 V to 3 V	0x1
	LSAD_RANGE_N1V_1V	Dynamic range is -1 V to 1 V	0x2
	LSAD_RANGE_0V_2V	Dynamic range is 0 V to 2 V	0x3
	LSAD_RANGE_1V_2V	Dynamic range is 1 V to 2 V	0x4
	LSAD_RANGE_2V_3V	Dynamic range is 2 V to 3 V	0x5
	LSAD_RANGE_N1V_0V	Dynamic range is -1 V to 0 V	0x6
	LSAD_RANGE_0V_1V	Dynamic range is 0 V to 1 V	0x7

8.4.1.2 A_DIV_CLK_CTRL Settings

Bit Field	Field Name	Field Description
7:4	DIV_CLK_CTRL_LSAD_FREQ_CTRL	Low speed A/D sampling rate prescaler

Field Name	Value Symbol	Value Description	Hex Value
DIV_CLK_CTRL_LSAD_FREQ_CTRL	LSAD_DISABLE	Disable LSAD converter	0x0
	LSAD_PRESCALE_40	Divide MCLK by 40	0x1
	LSAD_PRESCALE_60	Divide MCLK by 60	0x2
	LSAD_PRESCALE_80	Divide MCLK by 80	0x3
	LSAD_PRESCALE_100*	Divide MCLK by 100	0x4
	LSAD_PRESCALE_120	Divide MCLK by 120	0x5
	LSAD_PRESCALE_140	Divide MCLK by 140	0x6
	LSAD_PRESCALE_160	Divide MCLK by 160	0x7
	LSAD_PRESCALE_180	Divide MCLK by 180	0x8
	LSAD_PRESCALE_200	Divide MCLK by 200	0x9
	LSAD_PRESCALE_220	Divide MCLK by 220	0xA
	LSAD_PRESCALE_240	Divide MCLK by 240	0xB
	LSAD_PRESCALE_260	Divide MCLK by 260	0xC
	LSAD_PRESCALE_280	Divide MCLK by 280	0xD
	LSAD_PRESCALE_300	Divide MCLK by 300	0xE
	LSAD_PRESCALE_320	Divide MCLK by 320	0xF

8.5 PULSE CODE MODULATION (PCM) INTERFACE

The PCM interface allows digital audio to be transferred between Orela 4500 and another device (for instance, another Orela 4500 chip) that implements a PCM interface.

The PCM interface has four pins that are multiplexed with other functionality. The multiplexed PCM pins are as follows:

- A frame signal (PCM_FR)
- A serial input (PCM_IN)
- A serial output (PCM_OUT)
- A clock input (PCM_CLK)

For information on how to configure these multiplexed pins for PCM functionality refer to Section 8.1, "General Purpose I/O (GPIO)" on page 69.

The PCM interface is a synchronous interface that can be operated in both master and slave modes, where a PCM master is defined as the source of the frame signal on PCM_FR, and a PCM slave is a receiver of the frame signal on PCM_FR. The PCM_SLAVE bit field in the D_PCM_CFG

control register determines the master/slave configuration of the interface. The PCM interface is enabled using the `PCM_ENABLE` bit field in the `D_PCM_CFG` control register.

Note: A clock signal must be applied to the `PCM_CLK` pin to drive the data transmission/reception. The clock must be the same on both the master and slave sides, and the frequency of the clock shall not exceed `SYS_CLK`.

Data is transmitted and received in 32-bit frames, each composed of two 16-bit subframes. Each frame of data to be transmitted is stored into the two 16-bit registers `D_PCM_DATA0` and `D_PCM_DATA1` (each representing one of the subframes), where it will be automatically transferred to a PCM buffer for transmission. Similarly, each frame of received data may be read from the `D_PCM_DATA0` and `D_PCM_DATA1` registers.

While the data is being transmitted and received it may be buffered using a small FIFO accessible through the `D_PCM_DATA0` and `D_PCM_DATA1` registers or directly through the `D_PCM_DATA0_INDEX<X>` and `D_PCM_DATA1_INDEX<X>` buffer registers as shown in Figure 29. By using this buffer, the overhead for communication using the PCM interface will be reduced as the RCore will not need to service as many PCM interrupt requests.

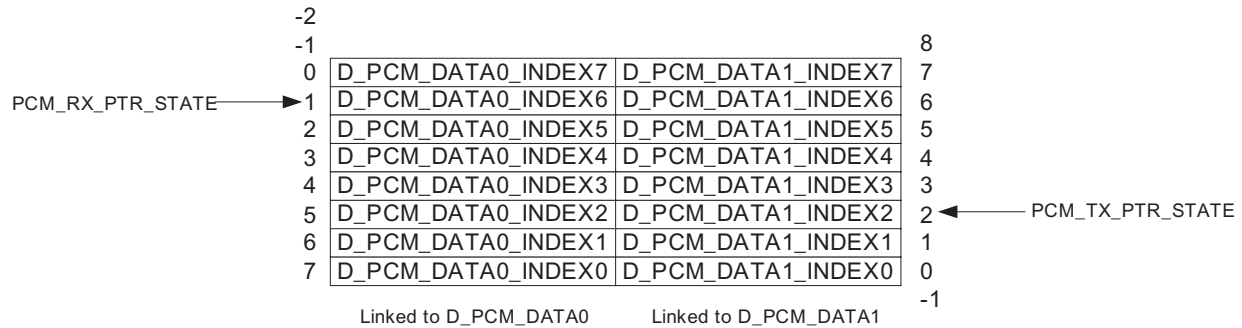


Figure 29: PCM interface buffer

The current state of reception buffering is indicated in the `PCM_RX_OVERFLOW` bit and the `PCM_RX_PTR_STATE` bit field of the `D_PCM_CTRL` register. The `PCM_RX_PTR_STATE` bit field provides a pointer to the next index of the buffer that will be read when reading from the registers `D_PCM_DATA0` and `D_PCM_DATA1`. It is automatically incremented when a data frame has been received and automatically decremented when a data item is read through the `D_PCM_DATA0` and `D_PCM_DATA1` registers. Since the buffer can hold a maximum of 8 elements, any value above 7 is invalid. Additionally, the value 0xF (or two's complement -1) indicates that the data buffer is still empty but there is a data frame that is in the process of being received. The value 0xE (or two's complement -2) indicates that not only is there no available data, there is no data that is currently being received and the receive subsystem is in its idle state. If the data buffer is full and a frame is received by the PCM interface before the RCore reads from the interface, the oldest frame of data will be discarded. As well, the `PCM_RX_OVERFLOW` sticky bit will be set and remain in that state until explicitly cleared by writing a value of 1 to the `PCM_RX_OVERFLOW` bit.

The `PCM_TX_UNDERFLOW` bit and `PCM_TX_PTR_STATE` bit field in the `D_PCM_CTRL` register indicate the current state of transmission buffering. The `PCM_TX_PTR_STATE` bit field provides a pointer to the next index that will be filled in the buffer when writing to `D_PCM_DATA0` and

D_PCM_DATA1. It is automatically decremented when a data frame has been transmitted, and automatically incremented when a data item is written to the D_PCM_DATA0 and D_PCM_DATA1 registers and as a result, is added to the buffer. Since the buffer can hold a maximum of 8 elements, any value above 8 is invalid. Writing to the buffer when it is in this state will result in the data that was written being ignored by the FIFO. The value 0x0 indicates that the data buffer is empty while the last data frame is being sent. The value 0xF (or two's complement -1) indicates that all valid data has been sent and until more data is added to the transmission buffer, the interface will resend the last valid data frame. This state may also be used to indicate that the transmission portion of the PCM interface is in an idle state. If the data buffer was empty and a non-valid frame was transmitted, the PCM_TX_UNDERFLOW sticky bit will be set and remain in that state until explicitly cleared by writing a value of 1 to the PCM_TX_UNDERFLOW bit.

When disabling the PCM interface, any data that has been loaded into the transmit buffer but has not been transmitted yet, is sent before the interface goes into its idle mode. In any case where the buffer is already empty, the interface goes into idle mode immediately. The buffer would be already empty if the interface is being used to receive data only, or the interface is to be disabled in response to an interrupt indicating the transmit buffer is empty. Otherwise the interface will go into idle mode following the completion of the transmission of the last frame of buffered data. In process, transmissions of words are always completed.

Note: Both the receive and transmit operations use the same buffer. Therefore received frames must be read from buffer locations that are to be written to before actually writing frames to be transmitted to the buffer. This will prevent frames that have been read from being overwritten by frames that are to be transmitted.

8.5.1 Interrupts

The PCM interface has an interrupt associated with the transmission and reception of values over the PCM interface. See Section 9.1, "Interrupt Controller" on page 101 for more information about interrupt configuration and handling.

Provided the transmit interrupt has been enabled by clearing the PCM_TX_INT bit in the D_PCM_CFG register, an interrupt will be generated by the Orela 4500 chip whenever the transmit buffer is empty, as indicated by a value of 0x0 (empty) or 0xF (all sent) in the PCM_TX_PTR_STATE bit field. This will result in an interrupt occurring following the transmission of each frame if no buffering is used.

Interrupts may also be used to indicate the reception of data through the PCM interface and are configured by setting the PCM_RX_INT_CFG bit field of the D_PCM_CFG register. An interrupt will be generated if the value of the PCM_RX_PTR_STATE bit field of the D_PCM_CTRL register equals or exceeds the value of this bit field, provided the interrupt has been enabled by clearing the PCM_RX_INT bit in the D_PCM_CFG register. Similar to the transmit interrupt case, the PCM_RX_INT_CFG bit field defaults to generating an interrupt after each frame, providing support for unbuffered operation by default.

The interrupt as seen by the interrupt controller is generated by performing a logical OR on interrupts generated for data transmission and interrupts generated for data reception.

8.5.2 Operation

When transmitting data over the PCM interface, the master controls when data is transferred. Transmission and reception of data proceeds one bit at a time starting one clock cycle after the

frame signal changes. It should be noted that other than the frame generation, both the master and slave share the same functionality. For details on the PCM data transmission format see Section 8.5.3, "Data Transfer Formatting Options" on page 83.

When a frame signal is generated by a PCM master or received by a PCM slave, the data elements in `D_PCM_DATA0_INDEX0` and `D_PCM_DATA1_INDEX0` are copied to an internal buffer for transmission.

Following the completion of a frame transfer, the data in the FIFO is shifted downwards with data from the internal receive buffer being inserted into data elements `D_PCM_DATA0_INDEX7` and `D_PCM_DATA1_INDEX7`. The following occurs at the same time:

1. If the transmit interrupt is enabled:
 - The `PCM_TX_PTR_STATE` bit field is decremented.
 - If required, the `PCM_TX_UNDERFLOW` bit is updated.
2. If the receive interrupt is enabled:
 - The `PCM_RX_PTR_STATE` bit field is incremented
 - If required, the `PCM_RX_OVERFLOW` bits is updated.

After the buffer and control register updates, an interrupt will be generated if either or both of the following conditions are met:

- Transmit interrupts are enabled and the transmit buffer is empty.
- Receive interrupts are enabled and the receive buffer has at least `PCM_RX_INT_CFG` available data elements.

For full-duplex transmission, both the transmit and receive interrupts should be enabled and configured for both the master and the slave. For half-duplex transmission only one of these interrupts needs to be configured on each side of the transmission. Typically, the master will need to enable and configure the transmit interrupt allowing the transmission of data and the slave will need to enable and configure the receive interrupt allowing the reception of data.

For single frame transfers the interface should be disabled following the completion of the frame transfer. As noted above, if the PCM interface is being used to only transmit data it may be disabled immediately following the write to the transmission buffer, as the interface will not go into idle mode until all of the data frames have been transmitted.

8.5.3 Data Transfer Formatting Options

The data transferred can be formatted in various ways. The descriptions below are from the viewpoint of the master; when in slave mode, an Orela 4500 PCM interface must be configured to match the format of the master.

Framing of 16-bit subframes or 32-bit frames

Frame signals may be transmitted on a per-frame basis or on a per-subframe basis. The choice is configured with the `PCM_SUBFRAME` bit in the `D_PCM_CFG` control register. The default is transmission of frame signals on a per-frame basis.

Note: When the interface is configured for per-subframe frame signals, interrupts on the slave are still only generated due to frame signals corresponding to the reception of full frames; every second frame signal on the slave is ignored, from the standpoint of interrupt generation. On the master, interrupts are

generated for every frame signal, regardless of whether the interface is configured for per-frame frame signal or per-subframe frame signal.

Bit order of transmitted words

Subframes may be transmitted beginning with the MSB of the subframe, or with the LSB of the subframe. The bit order is configured with the PCM_BIT_ORDER bit in the D_PCM_CFG control register. The default is transmission of the LSB of the subframe first.

Alignment of the frame signal with the frame or subframe

The frame signal may be sent with the rising edge occurring simultaneously with the start of the first transmitted bit of the frame or subframe (depending on whether frame signals are sent for frames or subframes, as described above), or with the falling edge occurring simultaneously with the end of the last transmitted bit of the frame or subframe. The alignment is configured with the PCM_FRAME_ALIGN bit in the D_PCM_CFG control register. The default is for the frame signal's falling edge to occur simultaneously with the end of the last transmitted bit.

Frame signal width The transmitted frame signal can be logical high for the duration of the transmission of a subframe, or it can be logical high for the duration of transmission of a single bit. The frame signal width is configured with the PCM_FRAME_WIDTH bit in the D_PCM_CFG control register.

Note: The generation of interrupts on the slave is not affected by the width of the frame signal. The default is for the frame signal to be logical high for the duration of a single bit.

Appendix B, "PCM Timing Diagrams" on page 131 provides a functional timing diagram for each of the possible data transfer modes.

8.5.4 Control and Configuration Registers

Register Name	Register Description	Address
D_PCM_DATA0	Subframe zero	Y:0x4002
D_PCM_DATA1	Subframe one	Y:0x4003
D_PCM_CTRL	PCM interface transmission control	Y:0x403F
D_PCM_DATA<X>_INDEX<Y>	Direct PCM Buffer access to data subframe X at data index Y	Y:0x4040 to Y:0x404F
D_PCM_CFG	PCM interface configuration	Y:0x8002

8.5.4.1 D_PCM_CTRL Settings

Bit Field	Field Name	Field Description
12	PCM_TX_UNDERFLOW_RESET	Setting this bit to one resets the transmit underflow flag
12	PCM_TX_UNDERFLOW	Sticky underflow indicator flag for PCM data transmission: 0x0: No transmit underflow has occurred 0x1: A transmit underflow has occurred
11:8	PCM_TX_PTR_STATE	Transmit FIFO pointer and state variable
4	PCM_RX_OVERFLOW_RESET	Setting this bit to one resets the receive overflow flag
4	PCM_RX_OVERFLOW	Sticky overflow indicator flag for PCM data reception: 0x0: No receive overflow has occurred 0x1: A receive overflow has occurred
3:0	PCM_RX_PTR_STATE	Receive FIFO pointer and state variable

Field Name	Value Symbol	Value Description	Hex Value
PCM_TX_PTR_STATE	PCM_TX_PTR_STATE_EMPTY	FIFO is empty, data is being sent	0x0
	PCM_TX_PTR_STATE_ALLSENT*	Idle state, all data has been sent	0xF
PCM_RX_PTR_STATE	PCM_RX_PTR_STATE_IDLE*	Idle state, no data has been received	0xE
	PCM_RX_PTR_STATE_1ST_RCV	Receiving the first two data words	0xF

8.5.4.2 D_PCM_CFG Settings

Bit Field	Field Name	Field Description
10:8	PCM_RX_INT_CFG	Select the number of frames to receive between receive interrupts
7	PCM_TX_INT	Enable/disable the transmit interrupt
6	PCM_RX_INT	Enable/disable the receive interrupt
5	PCM_FRAME_WIDTH	Select the width of the frame signal
4	PCM_FRAME_ALIGN	Select alignment of frame signal to the first bit or the last bit of the frame or subframe
3	PCM_BIT_ORDER	Select transmission of MSB first or LSB first
2	PCM_SUBFRAME	Select framing of 16-bit subframes or 32-bit frames
1	PCM_ENABLE	Enable/disable PCM interface
0	PCM_SLAVE	Master/slave mode configuration

Field Name	Value Symbol	Value Description	Hex Value
PCM_RX_INT_CFG	PCM_RX_INT_EVERY_1*	Generate interrupt every frame	0x0
	PCM_RX_INT_EVERY_2	Generate interrupt every 2 frames	0x1
	PCM_RX_INT_EVERY_3	Generate interrupt every 3 frames	0x2
	PCM_RX_INT_EVERY_4	Generate interrupt every 4 frames	0x3
	PCM_RX_INT_EVERY_5	Generate interrupt every 5 frames	0x4
	PCM_RX_INT_EVERY_6	Generate interrupt every 6 frames	0x5
	PCM_RX_INT_EVERY_7	Generate interrupt every 7 frames	0x6
	PCM_RX_INT_EVERY_8	Generate interrupt every 8 frames	0x7
PCM_TX_INT	PCM_ENABLE*	Enable the PCM transmit interrupt	0x0
	PCM_DISABLE	Disable the PCM transmit interrupt	0x1
PCM_RX_INT	PCM_ENABLE*	Enable the PCM receive interrupt	0x0
	PCM_DISABLE	Disable the PCM receive interrupt	0x1
PCM_FRAME_WIDTH	PCM_FRAME_WIDTH_SHORT*	Frame signal has single-bit duration	0x0
	PCM_FRAME_WIDTH_LONG	Frame signal has subframe duration	0x1
PCM_FRAME_ALIGN	PCM_FRAME_ALIGN_LAST_BIT*	Align frame signal to last bit	0x0
	PCM_FRAME_ALIGN_FIRST_BIT	Align frame signal to first bit	0x1
PCM_BIT_ORDER	PCM_BIT_ORDER_LSB_FIRST*	Transmit LSB first	0x0
	PCM_BIT_ORDER_MSB_FIRST	Transmit MSB first	0x1
PCM_SUBFRAME	PCM_SUBFRAME_DISABLE*	Frame 32-bit frame	0x0
	PCM_SUBFRAME_ENABLE	Frame 16-bit subframe	0x1
PCM_ENABLE	PCM_DISABLE*	Disable the PCM interface	0x0
	PCM_ENABLE	Enable the PCM interface	0x1
PCM_SLAVE	PCM_MASTER	Interface is PCM master	0x0
	PCM_SLAVE*	Interface is PCM slave	0x1

8.6 SPI PORT

The Serial Peripheral Interface (SPI) port conforms to the standard SPI, mode 0 or mode 2. One of the main uses is to communicate to external non-volatile storage devices such as EEPROM. Its operation is controlled and configured using the D_SPI_CTRL and D_SPI_CFG control registers. The SPI port is enabled in the D_SPI_CFG control register using the SPI_CFG_ENABLE bit field.

Transmitted/received data are stored in the D_SPI_DATA transmit and receive register.

The speed of the SPI port is configured relative to the speed of SYS_CLK by dividing SYS_CLK using the SPI_CFG_PRESCALE bit field from the D_SPI_CFG register. The clock for the SPI port may also be inverted by setting the SPI_CTRL_CLKPOL bit in the D_SPI_CTRL register. For more information regarding the configuration of SYS_CLK see Section 9.3, "Oscillator Circuitry and Clock Generation" on page 107.

When transferring data using the SPI port, the SPI_CTRL_RW bit of the D_SPI_CTRL register indicates whether data is to be written to or read from the SPI port. Writing a 0x1 to the

SPI_CTRL_START bit of the D_SPI_CTRL register starts a data transaction. This bit may be polled as SPI_CTRL_BUSY, which is cleared when the transaction is completed. The number of bits that should be sent in each transaction may be specified using the SPI_CTRL_N_BITS bit field of the D_SPI_CTRL register. Note that the SPI_CS_N bit of the D_SPI_CTRL register should be set to indicate whether a chip connected to the SPI interface should be selected.

The SPI port supports level translation that allows for access to an external EEPROM during normal system operation (i.e. when power-on reset, booting, etc. have finished). This allows access to the EEPROM even when the system is operating in low-voltage power supply mode without first having to switch into double-voltage power supply mode. Level translation is enabled using the PSU_CTRL_VSPI_MODE control bit in the A_PSU_CTRL register. For more information about setting power supply modes see Section 9.7, “Power Supply Regulation and Management” on page 121.

Note: To access the EEPROM using the level translation scheme described above, the external EEPROM must be supplied directly from VDBL (and not from VDDC, which is normally the case).

8.6.1 Interrupts

The SPI port is associated with the SPI interrupt. The SPI interrupt is generated when a transaction completes. If the transaction were a read from SPI, the new data are ready to read in the D_SPI_DATA register; if the transaction were a write to SPI, new data can be loaded into D_SPI_DATA. See Section 9.1, “Interrupt Controller” on page 101 for more information about interrupt configuration and handling.

8.6.2 Control and Configuration Registers

Register Name	Register Description	Address
A_PSU_CTRL	SPI port pin power source selection	A:0x1A
D_SPI_DATA	SPI receive/transmit register	Y:0x4000
D_SPI_CTRL	SPI port control	Y:0x4001
D_SPI_CFG	SPI port configuration	Y:0x8001

8.6.2.1 A_PSU_CTRL Settings

Bit Field	Field Name	Field Description
4	PSU_CTRL_VSPI_MODE	Select the voltage at which the SPI port will operate

Field Name	Value Symbol	Value Description	Hex Value
PSU_CTRL_VSPI_MODE	VSPI_MODE_VDDC*	SPI port operates at VDDC	0x0
	VSPI_MODE_VDBL	SPI port operates at VDBL (Level translated from VDDC)	0x1

8.6.2.2 D_SPI_CTRL Settings

Bit Field	Field Name	Field Description
7	SPI_CTRL_BUSY	Indicates whether a transaction is currently in progress 0x1: SPI port transaction in progress 0x0: SPI port idle
7	SPI_CTRL_START	Starts a SPI transaction when written with 0x1
6	SPI_CTRL_RW	Control read/write data from SPI
5	SPI_CTRL_CLKPOL	Set clock polarity
4	SPI_CS_N	Indicates the level of the chip select
3:0	SPI_CTRL_N_BITS	The number of bits to send (n+1, ranging from 1 to 16)

Field Name	Value Symbol	Value Description	Hex Value
SPI_CTRL_RW	SPI_WRITE*	Write data to SPI	0x0
	SPI_READ	Read data from SPI	0x1
SPI_CTRL_CLKPOL	SPI_CLKPOL_NORMAL*	Normal clock polarity	0x0
	SPI_CLKPOL_INVERSE	Inverse clock polarity	0x1
SPI_CS_N	SPI_CS_LOW*	Chip select low, active	0x0
	SPI_CS_HIGH	Chip select high, inactive	0x1

8.6.2.3 D_SPI_CFG Settings

Bit Field	Field Name	Field Description
3	SPI_CFG_ENABLE	Enable/disable SPI port
2:0	SPI_CFG_PRESCALE	Select prescale factor used to generate SPI clock

Field Name	Value Symbol	Value Description	Hex Value
SPI_CFG_ENABLE	SPI_PORT_DISABLE*	Disable SPI port	0x0
	SPI_PORT_ENABLE	Enable SPI port	0x1
SPI_CFG_PRESCALE	SPI_CLK_PRESCALE_2	SYS_CLK prescale of 2	0x0
	SPI_CLK_PRESCALE_4	SYS_CLK prescale of 4	0x1
	SPI_CLK_PRESCALE_8	SYS_CLK prescale of 8	0x2
	SPI_CLK_PRESCALE_16	SYS_CLK prescale of 16	0x3
	SPI_CLK_PRESCALE_32	SYS_CLK prescale of 32	0x4
	SPI_CLK_PRESCALE_64	SYS_CLK prescale of 64	0x5
	SPI_CLK_PRESCALE_128*	SYS_CLK prescale of 128	0x6
	SPI_CLK_PRESCALE_256	SYS_CLK prescale of 256	0x7

8.7 TWSS INTERFACE

The Two-Wire Synchronous Serial (TWSS) interface is an industry-standard interface for high-speed transmission of data between Orela 4500 and an external device. The interface operates at speeds up to 100 kbit/sec for system clocks (SYS_CLK) equal to 1.28 MHz. For system clock frequencies equal to or higher than 1.92 MHz the interface can operate at speeds up to 400 kbit/sec.

The TWSS interface can be controlled and configured with the D_TWSS_CTRL and D_TWSS_CFG control registers. Enabling/disabling is also done in the D_TWSS_CFG control register using the bit field TWSS_CFG_ENABLE. Data on the TWSS port is read and written through the D_TWSS_DATA register (one shared register for both receive and transmit). The TWSS interface operates in slave mode only.

8.7.1 Operation

8.7.1.1 Master to Slave Transfer Initiation

A data transfer is always initiated by the master by transmitting a START bit. When the slave receives the START bit, it sets the TWSS_CTRL_BUSY control bit to one. Subsequently, the master transmits a seven-bit slave address.

When a master wants to either receive or transmit data, the master can address a slave in two ways:

- A master can address a slave by transmitting a start bit followed by the general call address 0b0000000 and a R/W-bit (TWSS_CTRL_Rw). The general call address will always be recognized by the slave as a valid slave address. The slave's response depends on the

TwSS_CTRL_RW bit: if TwSS_CTRL_RW is zero (master wants to transmit data) the slave will acknowledge the reception. If TwSS_CTRL_RW is one (master wants to receive data) no acknowledge will be returned. Subsequently, the TwSS_CTRL_GEN_CALL control bit will be set to one to indicate that the slave was addressed by the general call address and the TwSS_CTRL_ADDR_RCV control bit will be set to one to indicate the reception of a valid slave address. An interrupt (TwSS interrupt) is generated upon reception of the slave address.

- A master can address a slave by transmitting a start bit followed by the known slave address (not the general call address) and a R/W-bit (TwSS_CTRL_RW). In that case the address will be recognized by the slave only if there is a match between the call address and the configured slave address (TwSS_CTRL_SLAVE_ID) on the slave. The slave will then acknowledge the reception of this address and subsequently a TwSS interrupt will be generated. The TwSS_CTRL_GEN_CALL bit will be set to zero to indicate that the slave was addressed using the specific slave address and not the general call address. The TwSS_CTRL_ADDR_RCV will be set to one upon reception of the slave address.

8.7.1.2 TWSS Receive Mode

When an external master wants to transmit data to a slave, it initiates the transmission as described in Section 8.7.1.1, “Master to Slave Transfer Initiation” on page 89. The master sets the TwSS_CTRL_RW bit to zero to indicate that it wants to write to the slave.

The reception of the TwSS_CTRL_RW bit causes a TwSS interrupt to be generated; the slave then responds to the TwSS interrupt. Here it can be detected that a slave address was received by polling the value of the TwSS_CTRL_ADDR_RCV control register bit. Depending on the setting of the TwSS_CFG_AUTO_CLK_RELEASE bit, either of two things can happen:

- If this bit is set to zero, the reception of data bytes from the master continues as soon as the D_TWSS_DATA register is read (although there is no valid data in the register at this point).
- If the bit is set to one, the reception of data bytes from the master is stalled until the TwSS_CTRL_CLK_RELEASE bit is set to one, regardless of whether the D_TWSS_DATA register has been read. This provides handshake functionality.

When the reception of more data has been enabled again after having serviced the interrupt generated after the slave address reception, an acknowledge will be returned to the master and a TwSS interrupt will be generated each time a new data value is ready in the D_TWSS_DATA register. Again, depending on the setting of the TwSS_CFG_AUTO_CLK_RELEASE bit, either of two things can happen in the ISR:

- If this bit is set to zero, the reception of more data from the master will continue as soon as the D_TWSS_DATA register is read.
- If the bit is set to one, the reception of more data from the master will be stalled until the TwSS_CTRL_CLK_RELEASE control bit has been set to one, regardless of whether the D_TWSS_DATA register has been read.

The master terminates a data transfer by transmitting a stop bit.

8.7.1.3 TWSS Transmit Mode

When an external master wants to receive data from a slave, it initiates the transmission as described in Section 8.7.1.1, “Master to Slave Transfer Initiation” on page 89. The master sets the TwSS_CTRL_RW bit to one to indicate that it wants to receive data from the slave.

The recognition of the slave address by the slave causes an acknowledge to be returned and an interrupt (TWSS) to be generated on the slave. Furthermore, the TWSS_CTRL_ADDR_RCV bit is set.

When the TWSS interrupt is generated, the slave enters the ISR. Here, it can be detected that a slave address was received by polling the value of the TWSS_CTRL_ADDR_RCV control register bit. Depending on the setting of the TWSS_CTRL_AUTO_CLK_RELEASE bit, either of two things can happen:

- If this bit is set to zero, the transmission of data bytes to the master starts as soon as the D_TWSS_DATA register is written.
- If the bit is set to one, the transmission of data to the master does not start until the TWSS_CTRL_CLK_RELEASE control bit is set to one, regardless of whether the D_TWSS_DATA register has been written.

For subsequent data transmissions an interrupt is generated every time a data byte has been transmitted to the master and the master has acknowledged the reception of the data byte. Thus, the interrupt indicates that a new data byte can be written to the D_TWSS_DATA register. Again, depending on the setting of the TWSS_CTRL_AUTO_CLK_RELEASE bit, two things can happen in the ISR:

- If this bit is set to zero, the transmission of more data to the master continues as soon as the D_TWSS_DATA register is written.
- If the bit is set to one, the transmission of more data to the master stalls until the TWSS_CTRL_CLK_RELEASE control bit is set to one, regardless of whether the D_TWSS_DATA register has been written.

When the master has received one or more data bytes from the slave, it can finalize the reception of data bytes by transmitting a Not Acknowledge (NACK) after the reception of the last data byte, followed by a stop bit. In that case, a TWSS interrupt is not generated upon reception of the NACK. The master can also choose to transmit an Acknowledge (ACK) after the reception of the last data byte, followed by a stop bit. In that case, a TWSS interrupt is generated.

8.7.2 Interrupts

The TWSS interface is associated with the TWSS interrupt. In addition to the events explained above that will generate a TWSS interrupt, the interrupt may also be configured to additionally produce an interrupt upon the reception of a stop bit by setting the TWSS_CFG_STOP_INT_EBL bit in the D_TWSS_CFG register. When an interrupt is generated as the result of a stop bit, the TWSS_CTRL_STOP_RCV bit of the D_TWSS_CTRL register is set. See Section 9.1, “Interrupt Controller” on page 101 for more information about interrupt configuration and handling.

8.7.3 Control and Configuration Registers

Register Name	Register Description	Address
D_TWSS_DATA	TWSS data read/write (bits 7:0)	Y:0x400C
D_TWSS_CTRL	TWSS control	Y:0x400D
D_TWSS_CFG	TWSS configuration	Y:0x8008

8.7.3.1 D_TWSS_CTRL Settings

Bit Field	Field Name	Field Description
4	TWSS_CTRL_STOP_RCV	When interrupts for stop bits are enabled, this indicates whether the interrupt is due to a stop bit or received data. 0x0: Interrupt due to reception of data. 0x1: Interrupt due to reception of stop bit.
3	TWSS_CTRL_ADDR_RCV	Indicates whether a data byte or a slave address was received from a master. 0x0: Data byte was received. 0x1: Slave address was received.
2	TWSS_CTRL_BUSY	Indicates whether the TWSS interface is busy or not. 0x0: Interface is idle. Set after a STOP condition on the interface 0x1: Interface is busy. Set after a START condition on the interface
1	TWSS_CTRL_GEN_CALL	Indicates whether the chip was addressed with its configured slave address or with the generic address. 0x0: The chip was addressed with its slave address (See the D_TWSS_CFG register) 0x1: The chip was addressed with the generic address (0b0000000).
0	TWSS_CTRL_RW	Read/write control by external master. The external master writes this bit. 0x0: master wants to write data. 0x1: master wants to read data.
0	TWSS_CTRL_CLK_REL	Controls transmission/reception by holding/releasing clock

Field Name	Value Symbol	Value Description	Hex Value
TWSS_CTRL_CLK_REL	TWSS_CLK_RELEASE	Used for manual handshaking. Clock is released.	0x1

8.7.3.2 D_TWSS_CFG Settings

Bit Field	Field Name	Field Description
13	TWSS_CFG_STOP_INT_EBL	Enable/disable interrupt on TWSS stop condition
8	TWSS_CFG_AUTO_CLK_RELEASE	Select auto or manual handshaking or "clock release"
7	TWSS_CFG_ENABLE	Enable/disable TWSS interface
6:0	TWSS_CFG_SLAVE_ID	Contains the Slave ID

Field Name	Value Symbol	Value Description	Hex Value
TWSS_CFG_STOP_INT_EBL	TWSS_STOP_INT_DISABLE*	Disable stop condition interrupt	0x0
	TWSS_STOP_INT_ENABLE	Enable stop condition interrupt	0x1
TWSS_CFG_AUTO_CLK_RELEASE	TWSS_ENABLE_AUTO_CLK_RELEASE*	Enable automatic clock release	0x0
	TWSS_DISABLE_AUTO_CLK_RELEASE	Disable automatic clock release	0x1
TWSS_CFG_ENABLE	TWSS_DISABLE*	Disable TWSS	0x0
	TWSS_ENABLE	Enable TWSS	0x1

8.8 DEBUG PORT UART

The debug port UART is used to provide a number of EDK tools using the Orela 4500 chip with a half-duplex asynchronous serial interface using the standard RS-232 transmission protocol. Data transfers for the debug port UART use a standard data format with one start bit, eight data bits and one stop bit to transfer data through the EXT3 system register. When using the debug port, it is not recommended to use this register for any other purpose.

Algorithms and intellectual property that are stored in the memory of the chip and on its EEPROM may be protected by restricting access to these elements of Orela 4500 as configured using the ACCESS_CFG_ACCESS_MODE bit of the D_ACCESS_CFG register. For more information on using the debug port UART to send data to Orela 4500 or the debug port protocols and algorithm protection see the *Communication Protocols Manual*.

The debug port may also be used to stream audio to the chip. Debug port audio streaming is configured through the DBG_AUDIO_MODE_CTRL field of the D_DBG_AUDIO_MODE_CTRL register. For more information about configuring the Orela 4500 chip to use audio streaming over the debug port see Section 8.8.3, "Audio Streaming" on page 94.

8.8.1 Baud Rate Synchronization

The interface baud rate of the debug port is automatically configured from PCLK to an appropriate frequency when the debug port interface is synchronized. Baud rates between PCLK/256 and PCLK/5 are configurable. The default baud rate of 115200 bps for the debug port should be automatically configured when operating under the default system clock and PCLK settings. For more information about configuring PCLK, see Section 9.3, "Oscillator Circuitry and Clock Generation" on page 107.

To allow the selection of an alternate communications speed, the debug port enters into baud rate synchronization mode when an unknown command is received. Following this transition into synchronization mode, a Sync_Baud_Rate character (ESC or 0x1B) should be sent at the desired speed and the port will use this character to adjust the communication speed to match. If successful, the debug port interface should be ready to receive commands at the newly configured baud rate. Problems with configuring the baud rate may arise when sending unknown characters if the system accidentally misinterprets unknown characters as known characters and does not switch into synchronization mode. For example, the start bit of 0xFF when it is sent at 9600 baud may be interpreted as two valid bytes when the debug port is operating at 115200, and as a result, the system will not switch into synchronization mode. To avoid this problem, several baud rate synchronization sequences that are known to be effective in configuring the interface using several common devices are provided in Table 12.

TABLE 12: KNOWN EFFECTIVE BAUD RATE SYNCHRONIZATION SEQUENCES

Connection Type	Target Speed	Baud Rate Synchronization Sequence
RS-232	9600	0x1B
	19200 and Higher	0xFF, 0x1B
HI-PRO	20481	0xFF, 0x1B, 0x1B
Microcard	9600	0x1B, 0x1B

If the debug port is not idle when attempting baud rate synchronization, it may be necessary to retry the entire process. This may occur if a debug port session is interrupted after sending a command that requires parameters. In this case, the unknown and Sync_Baud_Rate characters may be interpreted as parameters and become ineffective. The debug host can verify that baud rate synchronization has been successful by requesting the status of the system and waiting for a response with a short timeout. If a response is not received, the process should be repeated. For more information on how to request system status see the *Communication Protocols Manual*.

8.8.2 Interrupts

To avoid receiving EXT3_RX and EXT3_TX interrupts as a result of data transfers over the debug port UART, the EXT3_RX and EXT3_TX interrupts must be disabled while communicating using the debug port.

8.8.3 Audio Streaming

To reduce the number of connections that need to be made to the Orelia 4500 chip, the audio streaming feature allows one data pin of the debug port UART to be connected to an analog audio input. Doing this will allow data and analog audio to be sent to the chip using the same connection. By sending commands to the debug port, a communication system can force the DEBUG_TX or DEBUG_RX pin into a high-impedance state so that audio can be sent using the same connection without interference from the debug port. For more information on the audio streaming feature of the debug port see the *LLCOM Audio Extensions for NOAHlink Hardware and Software Reference*, distributed with the LLCOM library.

8.8.4 Control and Configuration Registers

Register Name	Register Description	Address
D_DBG_AUDIO_MODE_CTRL	Indicates the current audio streaming mode of the debug port	Y:0x4010
D_ACCESS_CFG	Configure algorithm protection and status byte behaviour	Y:0x800B

8.8.4.1 D_DBG_AUDIO_MODE_CTRL Settings

Bit Field	Field Name	Field Description
1:0	DBG_AUDIO_MODE_CTRL	Indicates the current audio streaming mode of the debug port

Field Name	Value Symbol	Value Description	Hex Value
DBG_AUDIO_MODE_CTRL	DBG_AUDIO_MODE_NORMAL*	Debug port is not configured for audio streaming	0x0
	DBG_AUDIO_MODE_HI_Z	Debug port is configured for high-impedance streaming on DEBUG_TX	0x1
	DBG_AUDIO_MODE_ANALOG	Debug port is configured for high-impedance streaming on DEBUG_RX	0x2
	DBG_AUDIO_MODE_PULLUP	Debug port is configured for “pull-up” mode as part of a transition from analog mode to normal mode.	0x3

8.8.4.2 D_ACCESS_CFG Settings

Bit Field	Field Name	Field Description
2	ACCESS_CFG_COMPATIBLE_MODE	Select whether the status byte on Orela 4500 should be SignaKlara 1 (SK1) compatible
0	ACCESS_CFG_ACCESS_MODE	Restrict/unrestrict access to Orela 4500

Field Name	Value Symbol	Value Description	Hex Value
ACCESS_CFG_COMPATIBLE_MODE	ACCESS_COMPATIBLE_MODE_DISABLE*	Normal status byte used	0x0
	ACCESS_COMPATIBLE_MODE_ENABLE	SignaKlara 1 (SK1) compatible status byte used	0x1
ACCESS_CFG_ACCESS_MODE	ACCESS_UNRESTRICTED	No restrictions on access to memory and EEPROM	0x0
	ACCESS_RESTRICTED*	Restricted/limited access to memory and EEPROM	0x1

8.9 SSP_CFG INTERFACE

The SSP_CFG interface is a serial communications interface for transferring the values of the analog block control and configuration registers (hereafter referred to as analog block registers) to the RCore, and for setting the values of the analog block registers from the RCore. As such, the

SSP_CFG interface is an internal interface with no associated I/O pads. Even though it is an externally inaccessible interface, it is important to understand because any interactions between the RCore and the analog blocks will be influenced by its properties.

Analog block registers are at most 8 bits in width; their names are all prefixed with A_ in this document and in the system `#include` files. Each register has an address within the analog blocks, as specified in this document.

All accesses to the analog block registers via the SSP_CFG interface are done through two internal address registers (pointers), also contained in the analog blocks, which are set to point to (i.e. contain the address of) the analog block register(s) of interest. These internal pointers are named STReg (Start Address Pointer Register) and CURReg (Current Address Pointer Register), and are referred to collectively as *analog block pointers*.

An SSP_CFG transfer operation is initiated by writing to the D_SSP_CFG_CTRL control register. The value one must be written to the SSP_CFG_CTRL_START bit to initiate a transfer on the interface. The SSP_CFG_CTRL_OP bit field specifies an operation code that determines whether the transfer is a read or a write, and indicates the operation to be performed on the analog block register or pointer. Finally, the SSP_CFG_CTRL_DATA bit field provides the 8 bits of data to be written, if the transfer is a write operation.

Transfers on the SSP_CFG interface take 16 MCLK cycles to complete. During the transfer, the SSP_CFG_CTRL_BUSY bit in the D_SSP_CFG_CTRL control register will read as one, indicating that the interface is busy performing the transfer. At other times, the SSP_CFG_CTRL_BUSY bit is zero, indicating that the interface is idle. It is important to wait until SSP_CFG_CTRL_BUSY is zero before attempting any SSP_CFG operations. After the transfer completes, the data have been read or written. If the transfer was a read operation, the SSP_CFG_CTRL_DATA bit field provides the data read.

The operations which may be specified in the SSP_CFG_CTRL_OP bit field are designed to support transfers of single analog block registers, or groups of analog block registers with increasing addresses. An operation may either access (read or write) the analog block pointers, or it may access an analog block register, in which case the analog block register accessed is the one pointed to by either STReg or CURReg.

For example, to read a single analog block register A, two SSP_CFG transfers are required:

1. The first transfer specifies an operation which writes the address of A to an analog block pointer. The data portion of the transfer (i.e. the SSP_CFG_CTRL_DATA bit field of D_SSP_CFG_CTRL) is the address of A.
2. The second transfer specifies an operation which reads the analog block register pointed to by the pointer. When this transfer completes, the data field of D_SSP_CFG_CTRL contains the value of analog block register A.

The operations which access analog block registers may also specify that the pointer is incremented prior to accessing the analog block register. These auto-increment operations support block transfers of groups of analog block registers, with one SSP_CFG transfer to set the starting address followed by an SSP_CFG transfer to access each analog block register in the group, while concurrently incrementing the analog block pointer. As a result, the overhead of transferring an address for each analog block register can be avoided when using this feature. The incrementing operations all increment the pointer prior to its use (pre-increment).

For example, to write new values to four analog block registers A, B, C and D whose addresses are consecutive, five SSP_CFG transfers are required:

1. As in the single-register example above, the first transfer writes the address of A to the analog block pointer(s).
2. The second transfer specifies an operation which writes the analog block register pointed to by the pointer, with no pre-increment, and the data value in the transfer is the desired new value for analog block register A.
- 3, 4, 5. The third to fifth transfers specify an operation which writes via the pointer, with a pre-increment of one. The data values in these transfers are the desired new values for analog block registers B, C and D, respectively.

An alternative approach here could use the address of A less one in the first transfer, followed by four transfers that write the analog block register values while pre-incrementing the pointer.

The analog block pointers STReg and CURReg are 6-bit registers. When the pointers are set using SSP_CFG transfer operations which write to them, the two MSBs of the 8-bit SSP_CFG_CTRL_DATA bit field must be zero. When CURReg is incremented using the pre-incrementing operations, its value wraps around from 0x3F to 0x0.

For convenience the macros Write_Areg and Read_AReg, which may be used to write or read from an analog register respectively, have been provided in *boss.inc*. For more information about using the SSP_CFG interface using these macros see the *Orela 4500 Firmware Reference Manual*.

8.9.1 Control and Configuration Registers

Register	Description	Address
D_SSP_CFG_CTRL	SSP_CFG interface control	Y:0x4004

8.9.1.1 D_SSP_CFG_CTRL Settings

Bit Field	Field Name	Field Description
15	SSP_CFG_CTRL_START	Setting this bit to one initiates a transfer
15	SSP_CFG_CTRL_BUSY	Status of SSP_CFG interface: 0x0: Transfer is in progress 0x1: Interface is idle
11:8	SSP_CFG_CTRL_OP	Specify operation direction and behaviour
7:0	SSP_CFG_CTRL_DATA	8-bit value to write or that has been read

Field Name	Value Symbol	Value Description	Hex Value
SSP_CFG_CTRL_OP	SSP_CFG_OP_WRITE_ADDR_ST_CUR*	Write STReg; CURReg = STReg	0x0
	SSP_CFG_OP_WRITE_DATA_ST	Write (STReg)	0x1
	SSP_CFG_OP_WRITE_DATA_CUR	Write (CURReg)	0x5
	SSP_CFG_OP_WRITE_DATA_CUR_INC1	CURReg = CURReg + 1; Write (CURReg)	0x3
	SSP_CFG_OP_WRITE_DATA_CUR_INC2	CURReg = CURReg + 2; Write (CURReg)	0x7
	SSP_CFG_OP_READ_ADDR_ST	Read STReg	0x8
	SSP_CFG_OP_READ_DATA_ST_CUR	Read (STReg); CURReg = STReg	0x9
	SSP_CFG_OP_READ_ADDR_CUR	Read CURReg	0xA
	SSP_CFG_OP_READ_DATA_CUR	Read (CURReg)	0xD
	SSP_CFG_OP_READ_DATA_CUR_INC1	CURReg = CURReg + 1; Read (CURReg)	0xB
	SSP_CFG_OP_READ_DATA_CUR_INC2	CURReg = CURReg + 2; Read (CURReg)	0xF

8.10 WIRELESS RECEIVER REMOTE CONTROL INTERFACE

The Orela 4500 chip includes a wireless receiver interface capable of receiving data coming through an infrared photo diode (or other source) on the AI_RC pin. The wireless interface converts the current from the infrared photo diode to a voltage and demodulates the incoming switched-carrier modulated signal. A dedicated UART transfers incoming samples to the REMOTE_DATA bit field in the D_REMOTE_DATA control register.

The operation of the wireless receiver is shown in Figure 30 (the use of a photo diode on the input is assumed).

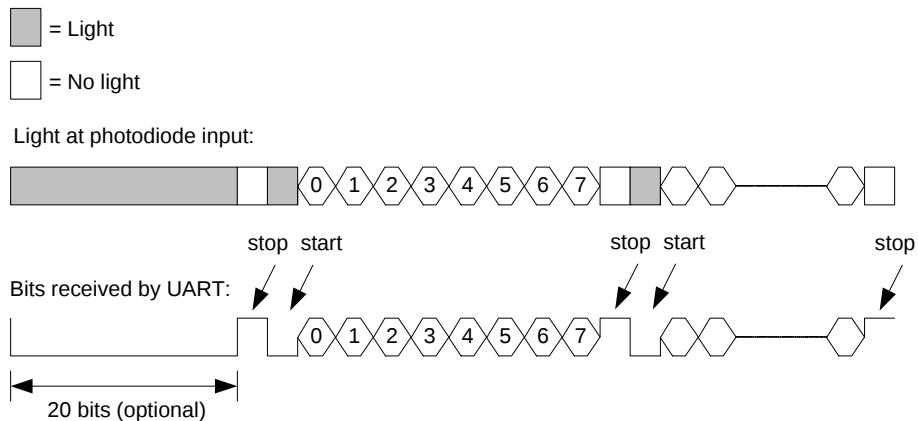


Figure 30: Wireless receiver remote control interface operation.

The reception of a sequence of RS-232-formatted, switched-carrier modulated bytes starts with the reception of a 20-bit burst sequence that ends with a stop bit. The burst sequence is followed by a start bit, a data byte, and a stop bit, which is again followed by a start bit and so on until all transmitted data bytes have been received. Optionally, the burst sequence can be disabled. This is

done in the D_REMOTE_CFG control register using the REMOTE_CFG_BURST_FILTER_ENABLE bit field.

Depending on the selected PCLK frequency, the speed of the UART must be configured so a baud rate of $1200 \pm 5\%$ bits/sec is obtained (at a 40-kHz carrier frequency). Thus, for various PCLK frequencies the remote control receiver needs to be configured so that the desired carrier frequency is always obtained. This is done by setting the REMOTE_CFG_PRESCALE bit field in the D_REMOTE_CFG control register to a value that satisfies the formula below (f_c is the carrier frequency that must be equal to 40 kHz):

$$f_c = \frac{f_{PCLK}}{((REMOTE_CFG_PRESCALE + 1) \cdot 8)}$$

In some applications, the initial data byte transferred from an external remote control to the chip will contain an identification code or similar device specifier, which makes it possible for the chip to determine whether it is being accessed by a valid remote control or not. Valid in this context means that the identification code transmitted by the remote control corresponds to an identification code stored on the chip. The REMOTE_DATA_INIT_BYTE_RCV bit in the D_REMOTE_DATA control register is set to one when the stop bit of the first data byte is received, regardless of whether the transfer was initiated with a burst sequence or not. The bit is automatically set to zero when the stop bit of the second data byte is received.

8.10.1 Interrupts

The remote control port has an associated interrupt. The REMOTE interrupt is set after successful reception of the stop bit associated with the reception of a data byte. See Section 9.1, "Interrupt Controller" on page 101 for information regarding interrupt configuration and handling.

8.10.2 Control and Configuration Registers

Register Name	Register Description	Address
D_REMOTE_DATA	Remote control data (bit 7:0)	Y:0x400E
D_REMOTE_CFG	Remote control configuration	Y:0x800C

8.10.2.1 D_REMOTE_DATA Settings

Bit Field	Field Name	Field Description
8	REMOTE_DATA_INIT_BYTE_RCV	Indicates that the first byte has been received 0x0: A byte that is not the first byte has been received 0x1: The first byte has been received
7:0	REMOTE_DATA	Received data value

8.10.2.2 D_REMOTE_CFG Settings

Bit Field	Field Name	Field Description
3	REMOTE_CFG_BURST_FILTER_ENABLE	Enable/disable remote burst filter. When enabled, the remote port expects an initial burst sequence; otherwise, it does not.
2:0	REMOTE_CFG_PRESCALE	Set the prescale value for the remote control port.

Field Name	Value Symbol	Value Description	Hex Value
REMOTE_CFG_BURST_FILTER_ENABLE	REMOTE_BURST_DISABLE	Disable remote burst	0x0
	REMOTE_BURST_ENABLE*	Enable remote burst	0x1
REMOTE_CFG_PRESCALE	REMOTE_DISABLE*	Disables the interface	0x0
	REMOTE_PRESCALE_2		0x1
	REMOTE_PRESCALE_3	Divide PCLK by 2	0x2
	REMOTE_PRESCALE_4	Divide PCLK by 3	0x3
	REMOTE_PRESCALE_5	Divide PCLK by 4	0x4
	REMOTE_PRESCALE_6	Divide PCLK by 5	0x5
	REMOTE_PRESCALE_7	Divide PCLK by 6	0x6
	REMOTE_PRESCALE_8	Divide PCLK by 7	0x7
		Divide PCLK by 8	

9.1 INTERRUPT CONTROLLER

The RCore DSP has a single interrupt channel that serves 13 interrupt sources in a prioritized manner. The interrupt controller also handles interrupt acknowledge flags. Each interrupt source has its own entry in the interrupt vector table. Furthermore, the priority scheme of the interrupt sources can be modified.

9.1.1 Interrupt Handling

The `D_INT_EBL` register contains the interrupt enable flags for each of the 13 possible interrupt sources. To permit interrupts to occur for a particular interrupt source, the bit in the `D_INT_EBL` register corresponding to that source must be set. The default configuration for `D_INT_EBL` is 0.

When an interrupt is generated, the interrupt status bit is set in the `D_INT_STATUS` register that corresponds to that interrupt source. The priority controller then checks to determine whether a higher priority interrupt is pending (which is indicated by the bit corresponding to a higher priority interrupt also being set). If not, and if the `D_INT_EBL` register bit corresponding to that source has been set, and if interrupts are globally enabled by `ST_IE` being set in the status register, then the interrupt is asserted and the RCore pushes the current PC on the system stack and jumps to the address in the interrupt vector table (see Section 7.2, “Program Memory” on page 64).

By enabling the Auto-acknowledge option, the RCore will then acknowledge the interrupt by resetting the bit in `D_INT_STATUS`. The interrupt controller can be configured to exhibit the following effects on servicing a given pending interrupt:

- Automatic rotation of interrupt system priority. See Section 9.1.4, “Automatic Priority Rotation” on page 103.
- Automatic clearing of the flag in the `D_INT_STATUS` register of the actual interrupt source. See Section 9.1.5, “Automatic Acknowledge” on page 103.

If the interrupt controller is configured to produce no effects when the RCore acknowledges the interrupt, it is expected that the RCore will complete all interrupt handling manually within the interrupt handling routine. This includes explicit clearing of flags in `D_INT_STATUS` and any desired priority setting update. Failure to properly manage interrupt handling will result in algorithm instability.

9.1.2 Interrupt Configuration

Three control registers determine the behaviour of the interrupt controller: `D_INT_CTRL`, `D_INT_STATUS` and `D_INT_EBL`. The configuration tables from Section 9.1.6, “Control and Configuration Registers” on page 103 show the bit functions in each of `D_INT_STATUS` and `D_INT_EBL` for read and write operations. The interrupt enable bit in the RCore status register (`ST` or `D_STATUS_REG`) should be cleared (i.e. set `ST_IE` to 0) before any changes are made to the `D_INT_EBL` register. Interrupts can be re-enabled by setting `ST_IE` to 1. The `SET_IE` instruction is provided for this purpose.

When an interrupt is handled by vectoring to an interrupt service routine, ST_IE is automatically set to zero. This is done to make interrupts uninterruptible. Thus, ST_IE must be set to one (via a SET_IE instruction) when the algorithm is prepared to again service other interrupts and before exiting an interrupt service routine.

The D_INT_STATUS register contains flags that show whether a particular interrupt is pending. A read access shows the status of each interrupt source (1 = interrupt pending). Writing 1 to the bit associated with a given interrupt source clears that bit, thereby acknowledging the interrupt. The clear operation is controlled by the data on each bit: 0 = leave flag unchanged and 1 = clear the flag. This arrangement allows any set of flags to be cleared with a single instruction.

The INT_CTRL_PRIO, INT_CTRL_PRIO_EBL, INT_EBL_AUTO_ROT, and INT_EBL_AUTO_ACK bit fields in the D_INT_CTRL register control the operation of the interrupt controller and are explained in Sections 9.1.3, "Interrupt Priority" on page 102 to 9.1.5, "Automatic Acknowledge" on page 103.

9.1.3 Interrupt Priority

The value in the INT_CTRL_PRIO_EBL bit position of the D_INT_CTRL register determines whether multiple interrupt priority schemes are accessible. When the INT_CTRL_PRIO_EBL bit is cleared only the default priority scheme is accessible; otherwise, all priority schemes are available.

If priority schemes, other than the default priority, are made available by setting the INT_CTRL_PRIO_EBL bit, the priority scheme for the system interrupts is selectable through the INT_CTRL_PRIO bit field in the D_INT_CTRL register. The value of this field allows for the selection of any of the interrupt priority schemes presented in Table 13. The numbers in this table correspond to the position of each interrupt that may be enabled in the D_INT_EBL register.

TABLE 13: INTERRUPT PRIORITY SETTINGS

INT_CTRL_PRIO	Interrupt priority order												
0000 (default)	0	1	2	3	4	5	6	7	8	9	10	11	12
0001	1	2	3	4	5	6	7	8	9	10	11	12	0
0010	2	3	4	5	6	7	8	9	10	11	12	0	1
0011	3	4	5	6	7	8	9	10	11	12	0	1	2
0100	4	5	6	7	8	9	10	11	12	0	1	2	3
0101	5	6	7	8	9	10	11	12	0	1	2	3	4
0110	6	7	8	9	10	11	12	0	1	2	3	4	5
0111	7	8	9	10	11	12	0	1	2	3	4	5	6
1000	8	9	10	11	12	0	1	2	3	4	5	6	7
1001	9	10	11	12	0	1	2	3	4	5	6	7	8
1010	10	11	12	0	1	2	3	4	5	6	7	8	9
1011	11	12	0	1	2	3	4	5	6	7	8	9	10
1100	12	0	1	2	3	4	5	6	7	8	9	10	11
Other	0	1	2	3	4	5	6	7	8	9	10	11	12

Reading the value of the INT_CTRL_PRI0 bit field will return the current active priority scheme. That is, zeros will be returned if the INT_CTRL_PRI0_EBL flag in the D_INT_CTRL register is set to 0, and the value present in the INT_CTRL_PRI0 bit field will be returned otherwise.

9.1.4 Automatic Priority Rotation

The INT_CTRL_AUTO_ROT bit in the D_INT_CTRL register enables automatic priority rotation. When automatic priority rotation is enabled, the value of INT_CTRL_PRI0 will be incremented modulo 0xD after each interrupt acknowledge. Over an extended period of time, this gives every interrupt source the same average priority and guaranteed interrupt service for all sources.

9.1.5 Automatic Acknowledge

The INT_CTRL_AUTO_ACK bit in the D_INT_CTRL register enables the automatic acknowledge feature. When an interrupt is acknowledged after vectoring to an interrupt service routine, the interrupt handler automatically clears the corresponding bit in the D_INT_STATUS register.

9.1.6 Control and Configuration Registers

Register Name	Register Description	Address
D_INT_CTRL	Interrupt control and configuration register	Y:400F
D_INT_STATUS	Interrupt status register	EXT5 (RCore register)
D_INT_EBL	Interrupt enable register	EXT6 (RCore register)

9.1.6.1 D_INT_CTRL Settings

Bit Field	Field Name	Field Description
15:12	INT_CTRL_PRI0	Select interrupt priority
2	INT_CTRL_PRI0_EBL	Enable interrupt priority settings
1	INT_CTRL_AUTO_ROT	Enable auto rotation of interrupt priorities
0	INT_CTRL_AUTO_ACK	Enable auto acknowledge of interrupts

Field Name	Value Symbol	Value Description	Hex Value
INT_STATUS_PRIORITY	INT_Prio_WOLA_DONE*	WOLA_DONE interrupt top priority	0x0
	INT_Prio_IO_BLOCK_FULL	IO_BLOCK_FULL interrupt top priority	0x1
	INT_Prio_PCM	PCM interrupt top priority	0x2
	INT_Prio_UART_RX	UART_RX interrupt top priority	0x3
	INT_Prio_UART_TX	UART_TX interrupt top priority	0x4
	INT_Prio_TIMER	TIMER interrupt top priority	0x5
	INT_Prio_WATCHDOG	WATCHDOG interrupt top priority	0x6
	INT_Prio_SPI	SPI interrupt top priority	0x7
	INT_Prio_TWSS	TWSS interrupt top priority	0x8
	INT_Prio_REMOTE	REMOTE interrupt top priority	0x9
	INT_Prio_EXT3_RX	EXT3_RX interrupt top priority	0xA
	INT_Prio_EXT3_TX	EXT3_TX interrupt top priority	0xB
	INT_Prio_GPIO	GPIO interrupt top priority	0xC
INT_CTRL_Prio_EBL	INT_PRIORITY_DISABLE*	Disable interrupt priority settings	0x0
	INT_PRIORITY_ENABLE	Enable interrupt priority settings	0x1
INT_CTRL_AUTO_ROT	INT_AUTO_ROT_DISABLE*	Disable auto rotation	0x0
	INT_AUTO_ROT_ENABLE	Enable auto rotation	0x1
INT_CTRL_AUTO_ACK	INT_AUTO_ACK_DISABLE*	Disable auto acknowledge	0x0
	INT_AUTO_ACK_ENABLE	Enable auto acknowledge	0x1

9.1.6.2 D_INT_STATUS Settings

Bit Field	Field Name	Field Description
12:0	INT_<interrupt>	A read of a specific bit returns the status of the associated interrupt Writing a 1 to a specific bit manually acknowledges the associated interrupt and clears the corresponding bit in the D_INT_STATUS control register

Field Name	Value Symbol	Value Description	Hex Value
INT_<interrupt>	INT_ACK_WOLA_DONE	Ack. WOLA_DONE interrupt (write)	0x1
	INT_ACK_IO_BLOCK_FULL	Ack. IO_BLOCK_FULL interrupt (write)	0x1
	INT_ACK_PCM	Ack. PCM interrupt (write)	0x1
	INT_ACK_UART_RX	Ack. UART_RX interrupt (write)	0x1
	INT_ACK_UART_TX	Ack. UART_TX interrupt (write)	0x1
	INT_ACK_TIMER	Ack. TIMER interrupt (write)	0x1
	INT_ACK_WATCHDOG	Ack. WATCHDOG interrupt (write)	0x1
	INT_ACK_SPI	Ack. SPI interrupt (write)	0x1
	INT_ACK_TWSS	Ack. TWSS interrupt (write)	0x1
	INT_ACK_REMOTE	Ack. REMOTE interrupt (write)	0x1
	INT_ACK_EXT3_RX	Ack. EXT3_RX interrupt (write)	0x1
	INT_ACK_EXT3_TX	Ack. EXT3_TX interrupt (write)	0x1
	INT_ACK_GPIO	Ack. GPIO interrupt (write)	0x1

9.1.6.3 D_INT_EBL Settings

Bit Field	Field Name	Field Description
12:0	INT_<interrupt>	Enable interrupt sources

Field Name	Value Symbol	Value Description	Hex Value
INT_<interrupt>	INT_EBL_WOLA_DONE	Disable WOLA_DONE interrupt	0x0
		Enable WOLA_DONE interrupt	0x1
	INT_EBL_IO_BLOCK_FULL	Disable IO_BLOCK_FULL interrupt	0x0
		Enable IO_BLOCK_FULL interrupt	0x1
	INT_EBL_PCM	Disable PCM interrupt	0x0
		Enable PCM interrupt	0x1
	INT_EBL_UART_RX	Disable UART_RX interrupt	0x0
		Enable UART_RX interrupt	0x1
	INT_EBL_UART_TX	Disable UART_TX interrupt	0x0
		Enable UART_TX interrupt	0x1
	INT_EBL_TIMER	Disable TIMER interrupt	0x0
		Enable TIMER interrupt	0x1
	INT_EBL_WATCHDOG	Disable WATCHDOG interrupt	0x0
		Enable WATCHDOG interrupt	0x1
	INT_EBL_SPI	Disable SPI interrupt	0x0
		Enable SPI interrupt	0x1
	INT_EBL_TWSS	Disable TWSS interrupt	0x0
		Enable TWSS interrupt	0x1
	INT_EBL_REMOTE	Disable REMOTE interrupt	0x0
		Enable REMOTE interrupt	0x1
	INT_EBL_EXT3_RX	Disable EXT3_RX interrupt	0x0
		Enable EXT3_RX interrupt	0x1
	INT_EBL_EXT3_TX	Disable EXT3_TX interrupt	0x0
		Enable EXT3_TX interrupt	0x1
	INT_EBL_GPIO	Disable GPIO interrupt	0x0
		Enable GPIO interrupt	0x1

9.2 BATTERY MONITOR

A battery monitor signals when the supply voltage to Orela 4500 has fallen below a specified threshold value set in the A_SUP_THRSH_DATA control register. The battery monitor was designed to tally the number of times the battery voltage level drops below a configurable threshold. The threshold value is adjustable from 0 V to 2 V in increments of 7.8 mV (from 0 to 255). The battery monitor will count up to a maximum of 255 where the counter will remain regardless of further drops below the threshold voltage until reset manually.

If the battery voltage level falls below the threshold and does not recover (i.e. it stays below the threshold voltage), the count increments every 80 ms (with MCLK=1.28 MHz). After the battery voltage recovers, the counter stops incrementing (but stays at the actual value) until the battery voltage falls below the threshold again.

At any time, the counter value can be read through the A_SUP_COUNT_DATA control register, and following the read, the value of the counter is automatically reset.

9.2.1 Control and Configuration Registers

Register Name	Register Description	Address
A_SUP_THRSH_DATA	Detection threshold, programmable from 0 V to 2 V in increments of 7.8 mV.	A:0x14
A_SUP_COUNT_DATA	Counter value	A:0x15

9.3 OSCILLATOR CIRCUITRY AND CLOCK GENERATION

The base clock for all operations on the Orela 4500 chip is the system clock (SYS_CLK). This clock may be acquired from one of three sources: the main on-chip oscillator, the system standby clock or an external clock signal. In turn, the SYS_CLK signal is used to define five other clocks, including three internal clock domains and two external clock outputs. A brief summary of the internal clock domains and the system blocks that make use of each clock domain can be found in Table 14. Figure 31 on page 109 shows the clock generation tree, providing reference as to how the various clock domains are related.

TABLE 14: CLOCK DOMAINS

Clock Domain	Affected Blocks
SYS_CLK	RCore IOP Memory GPIO interface PCM interface SPI port TWSS interface
WOLACLK	WOLA filterbank coprocessor

TABLE 14: CLOCK DOMAINS (CONTINUED)

Clock Domain	Affected Blocks
MCLK	A/D converters D/A converters Decimation filters Interpolation filters direct digital outputs SSP_CFG interface Low-speed A/D
PCLK	UART Debug port I ² S interface Remote control Timer Watchdog

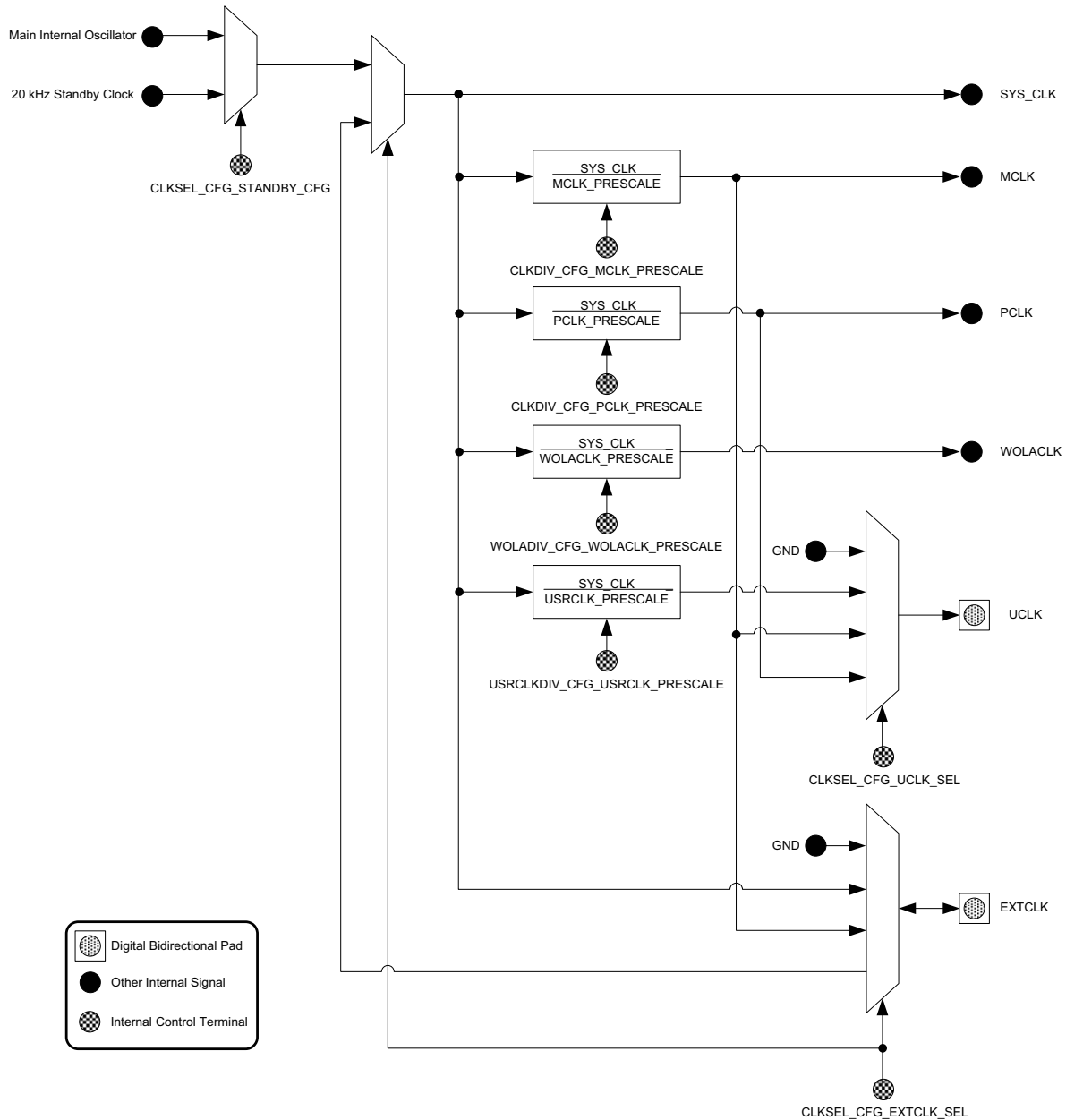


Figure 31: Clock derivations and relationships

9.3.1 On-chip Oscillator Configuration and Calibration

The on-chip oscillator frequency is configured by using the `CLK_CTRL_SYS_CLK` bit field from the `A_CLK_CTRL` control register to coarsely select a base frequency. This frequency may be adjusted to obtain an exact clock frequency if the oscillator is calibrated by changing the calibration value in the `A_CCR_DATA` control register. The default value in this control register is 0x80, which

allows 128 calibration steps in each direction. The calibration value 0x00 corresponds to the lowest frequency (approximately 20% lower than the frequency associated with a calibration value of 0x80) and the value 0xFF corresponds to the highest frequency (approximately 20% higher than the frequency associated with a calibration value of 0x80).

9.3.2 Clocks and Clock Domains

System Clock

The system clock (SYS_CLK) may be derived from the on-chip main oscillator for normal operation, the on-chip standby oscillator for the soft power down mode, or driven by an external source through the EXT_CLK pin. To select whether SYS_CLK should be clocked from an external clock source, the EXT_CLK pin should be configured as an input using the CLKSEL_CFG_EXTCLK_SEL bit field of the D_CLKSEL_CFG register. If the system is not configured for external clocking, the source for SYS_CLK may be selected from the on-chip oscillators using the CLKSEL_CFG_STANDBY_CLK pin from the D_CLKSEL_CFG register.

SYS_CLK is always set to the nominal frequency 1.80 MHz using the on-chip oscillator when Orela 4500 is reset. Switching to another internally generated (calibrated) clock frequency or an external clock source can only be done after the boot part of the Program ROM has finished execution. Although the internal oscillator for Orela 4500 can be configured for up to 13.2 MHz, operation at frequencies above 5.12 MHz is not characterized. As a result, only operational frequencies at or below this value will be supported.

When operating in soft power down mode, the standby clock oscillator can be used to drive SYS_CLK. This clock operates at approximately 20 kHz and helps to provide an ultra low power operation mode for Orela 4500. For more details about the standby clock oscillator and soft power down mode see Section 9.7, “Power Supply Regulation and Management” on page 121.

Main Clock

The main clock (MCLK) is derived from SYS_CLK by dividing SYS_CLK by the CLKDIV_CFG_MCLK_PRESCALE division factor specified in the D_CLKDIV_CFG control register. When performing the division, SYS_CLK will be divided by $(\text{CLKDIV_CFG_MCLK_PRESCALE} + 1)$ to obtain MCLK. For example if SYS_CLK has a frequency of 2.56 MHz and a MCLK frequency of 1.28 MHz is desired, CLKDIV_CFG_MCLK_PRESCALE should be set to 1 to obtain a division factor of 2. It should be noted that the behaviour for MCLK has only been characterized for frequencies between 100 kHz and 3.84 MHz. Only frequencies within this range are supported.

If an MCLK frequency exceeding 1.5 MHz is desired, the ADC_CUR_CTRL_HIGH_FREQ_EBL bit in the A_ADC_CUR_CTRL control register should be set to ensure proper current control for A/D converters. For more information regarding current configuration for A/D converters, see Section 3.6, “A/D Converters” on page 11.

Peripheral Clock

The peripheral clock (PCLK) is derived from SYS_CLK by dividing SYS_CLK by the CLKDIV_CFG_PCLK_PRESCALE division factor specified in the D_CLKDIV_CFG control register. Similarly to MCLK, SYS_CLK will be divided by $(\text{CLKDIV_CFG_PCLK_PRESCALE} + 1)$ to obtain the value of PCLK. It should be noted that the behaviour for PCLK has only been characterized for frequencies between 100 kHz and 1.92 MHz. As a result, only PCLK frequencies within this range are supported.

WOLA Clock

The WOLA clock (WOLACLK) is derived from SYS_CLK using the WOLADIV_CFG_WOLACLK_PRESCALE scaling factor specified in the D_WOLADIV_CFG control register in the following equation:

$$WOLACLK = \frac{8 \times SYS_CLK}{(WOLADIV_CFG_WOLACLK_PRESCALE + 8)}$$

The WOLACLK allows WOLA computations to occur at a slower frequency than they would when executing at SYS_CLK in order to spread the computation time over a larger time interval.

User Clock

The user clock (UCLK) output pin may be configured using the CLKSEL_CFG_UCLK_SEL bit field of the D_CLK_SEL_CFG control register. This bit field can configure UCLK to be disabled, to output either MCLK or PCLK, or to provide a specific user clock domain as output. If the output is configured to use the user specific clock domain, the clock signal will be connected to the USRCLK signal which is generated from SYS_CLK by dividing SYS_CLK by the USRCLKDIV_CFG_USRCLK_PRESCALE division factor specified in the D_USRCLKDIV_CFG control register. Similar to both MCLK and PCLK, SYS_CLK will be divided by (USRCLKDIV_CFG_USRCLK_PRESCALE + 1) to obtain the value of UCLK in this mode.

External Clock

The external clock (EXTCLK) input/output pin may be configured using the CLKSEL_CFG_EXTCLK_SEL bit field of the D_CLK_SEL_CFG control register. This bit field may be used to configure EXTCLK as disabled, as an output providing either SYS_CLK or MCLK, or as an input allowing an external oscillator to supply an arbitrary frequency clock to SYS_CLK.

9.3.3 Sampling Frequency Configuration

The (effective downsampled) sampling frequency f_s , expressed in Hz for both A/D converters and both D/A converters, depends on the frequency of MCLK and the value of the ADC_CTRL_SAMPLE_FREQ bit field in the A_ADC_CTRL control register related through the following equation:

$$f_s = \frac{f_{MCLK}}{(ADC_CTRL_SAMPLE_FREQ + 8) \cdot 8}$$

For example, if an MCLK frequency of 1.28 MHz is used and a sampling frequency of 16 kHz is desired, the value in the ADC_CTRL_SAMPLE_FREQ bit field should be set to 0x2. The selected sampling frequency is always the same for all converters. They cannot be selected independently for each converter. For a comprehensive listing of MCLK frequencies versus selectable sampling frequencies, see Appendix A, "Sampling Frequencies" on page 129.

For convenience, the macro Calc_Sample_Freq_Val has been provided in *boss.inc* to be used in selecting the value that ADC_CTRL_SAMPLE_FREQ should hold for a given MCLK frequency to approximate a desired sampling frequency. For more information about configuring the sampling frequency using Calc_Sample_Freq_Val see the *Orela 4500 Firmware Reference Manual*.

9.3.4 Control and Configuration Registers

Register Name	Register Description	Address
A_CCR_DATA	Clock calibration value	A:0x10
A_CLK_CTRL	System clock control	A:0x11
A_ADC_CTRL	Sampling frequency control	A:0x1D
D_CLKSEL_CFG	Clock source selection and configuration	Y:0x8009
D_CLKDIV_CFG	Clock division configuration for PCLK and MCLK	Y:0x800E
D_USRCLKDIV_CFG	Clock division configuration for USRCLK	Y:0x800F
D_WOLADIV_CFG	Clock division configuration for WOLACLK	Y:0x8010
D_STANDBY_CFG	20 kHz (approximately) power management clock control	Y:0x8011

9.3.4.1 A_CCR_DATA Settings

The A_CCR_DATA control register is documented in Section 9.3.1, “On-chip Oscillator Configuration and Calibration” on page 109.

9.3.4.2 A_CLK_CTRL Settings

Bit Field	Field Name	Field Description
3:0	CLK_CTRL_SYS_CLK	Select system clock frequency

Field Name	Value Symbol	Value Description	Hex Value
CLK_CTRL_SYS_CLK ¹	SYS_CLK_OFF	Turn off internal clock	0x0
	SYS_CLK_0M62	Select 0.62 MHz system clock	0x1
	SYS_CLK_0M92	Select 0.92 MHz system clock	0x2
	SYS_CLK_1M22	Select 1.22 MHz system clock	0x3
	SYS_CLK_1M51	Select 1.51 MHz system clock	0x4
	SYS_CLK_1M80*	Select 1.80 MHz system clock	0x5
	SYS_CLK_2M36	Select 2.36 MHz system clock	0x6
	SYS_CLK_2M91	Select 2.91 MHz system clock	0x7
	SYS_CLK_3M44	Select 3.44 MHz system clock	0x8
	SYS_CLK_4M46	Select 4.46 MHz system clock	0x9
	SYS_CLK_5M43	Select 5.43 MHz system clock	0xA
	SYS_CLK_6M78	Select 6.78 MHz system clock	0xB
	SYS_CLK_8M43	Select 8.43 MHz system clock	0xC
	SYS_CLK_10M2	Select 10.2 MHz system clock	0xD
	SYS_CLK_11M9	Select 11.9 MHz system clock	0xE
	SYS_CLK_13M2	Select 13.2 MHz system clock	0xF

1. The coarse frequency values for SYS_CLK as noted here are approximate and the actual SYS_CLK frequency value may vary.

9.3.4.3 A_ADC_CTRL Settings

Bit Field	Field Name	Field Description
5:0	ADC_CTRL_SAMPLE_FREQ	Select sampling frequency

Field Name	Value Symbol	Value Description	Hex Value
ADC_CTRL_SAMPLE_FREQ ¹	SAMPLE_FREQ_8KHZ_CLK_0M64	Sampling Freq. 8 kHz, MCLK 640 kHz	0x02
	SAMPLE_FREQ_20KHZ_CLK_1M28	Sampling Freq. 20 kHz, MCLK 1.28 MHz	0x00
	SAMPLE_FREQ_16KHZ_CLK_1M28	Sampling Freq. 16 kHz, MCLK 1.28 MHz	0x02
	SAMPLE_FREQ_8KHZ_CLK_1M28	Sampling Freq. 8 kHz, MCLK 1.28 MHz	0x0C
	SAMPLE_FREQ_20KHZ_CLK_1M92	Sampling Freq. 20 kHz, MCLK 1.92 MHz	0x04
	SAMPLE_FREQ_16KHZ_CLK_1M92	Sampling Freq. 16 kHz, MCLK 1.92 MHz	0x07
	SAMPLE_FREQ_8KHZ_CLK_1M92	Sampling Freq. 8 kHz, MCLK 1.92 MHz	0x16
	SAMPLE_FREQ_32KHZ_CLK_2M56	Sampling Freq. 32 kHz, MCLK 2.56 MHz	0x02
	SAMPLE_FREQ_20KHZ_CLK_2M56	Sampling Freq. 20 kHz, MCLK 2.56 MHz	0x08
	SAMPLE_FREQ_16KHZ_CLK_2M56	Sampling Freq. 16 kHz, MCLK 2.56 MHz	0x0C
	SAMPLE_FREQ_8KHZ_CLK_2M56	Sampling Freq. 8 kHz, MCLK 2.56 MHz	0x20
	SAMPLE_FREQ_32KHZ_CLK_3M84	Sampling Freq. 32 kHz, MCLK 3.84 MHz	0x07
	SAMPLE_FREQ_20KHZ_CLK_3M84	Sampling Freq. 20 kHz, MCLK 3.84 MHz	0x10
	SAMPLE_FREQ_16KHZ_CLK_3M84	Sampling Freq. 16 kHz, MCLK 3.84 MHz	0x16
	SAMPLE_FREQ_8KHZ_CLK_3M84	Sampling Freq. 8 kHz, MCLK 3.84 MHz	0x34

1. Other sampling frequencies are configurable using scaling factors of up to 54 (0x36) for ADC_CTRL_SAMPLE_FREQ, but do not have associated #define statements. Obtaining the MCLK values listed in this table require the system clock to be calibrated.

9.3.4.4 D_CLK_SEL_CFG Settings

Bit Field	Field Name	Field Description
6	CLKSEL_CFG_STANDBY_CLK	Select the internal oscillator source for SYS_CLK
5:4	CLKSEL_CFG_EXTCLK_SEL	Select the source and behaviour of EXTCLK
1:0	CLKSEL_CFG_UCLK_SEL	Select the source of UCLK

Field Name	Value Symbol	Value Description	Hex Value
CLKSEL_CFG_STANDBY_CFG	CLK_SYS_CLK_SELECT_DEFAULT*	Select default SYS_CLK behaviour	0x0
	CLK_SYS_CLK_SELECT_STANDBY	Select the standby clock	0x1
CLKSEL_CFG_EXTCLK_SEL	CLK_EXTCLK_DISABLE*	Disable external clock I/O	0x0
	CLK_EXTCLK_OUT_SYSClk_ENABLE	Deliver SYS_CLK output	0x1
	CLK_EXTCLK_OUT_MCLK_ENABLE	Deliver MCLK output	0x2
	CLK_EXTCLK_IN_ENABLE	Select external clock input	0x3
CLKSEL_CFG_UCLK_SEL	CLK_UCLK_DISABLE*	Disable user clock output	0x0
	CLK_UCLK_SEL_USRCLK	Deliver USRCLK output	0x1
	CLK_UCLK_SEL_MCLK	Deliver MCLK output	0x2
	CLK_UCLK_SEL_PCLK	Deliver PCLK output	0x3

9.3.4.5 D_CLKDIV_CFG Settings

Bit Field	Field Name	Field Description
14:8	CLKDIV_CFG_PCLK_PRESCALE	Set prescale factor used to generate PCLK
6:0	CLKDIV_CFG_MCLK_PRESCALE	Set prescale factor used to generate MCLK

Field Name	Value Symbol	Value Description	Hex Value
CLKDIV_CFG_PCLK_PRESCALE ¹	CLK_PCLK_PRESCALE_1*	Divide SYS_CLK by 1	0x0
	CLK_PCLK_PRESCALE_2	Divide SYS_CLK by 2	0x1
	CLK_PCLK_PRESCALE_3	Divide SYS_CLK by 3	0x2
	CLK_PCLK_PRESCALE_4	Divide SYS_CLK by 4	0x3
	CLK_PCLK_PRESCALE_5	Divide SYS_CLK by 5	0x4
	CLK_PCLK_PRESCALE_6	Divide SYS_CLK by 6	0x5
	CLK_PCLK_PRESCALE_7	Divide SYS_CLK by 7	0x6
	CLK_PCLK_PRESCALE_8	Divide SYS_CLK by 8	0x7
CLKDIV_CFG_MCLK_PRESCALE ¹	CLK_MCLK_PRESCALE_1*	Divide SYS_CLK by 1	0x0
	CLK_MCLK_PRESCALE_2	Divide SYS_CLK by 2	0x1
	CLK_MCLK_PRESCALE_3	Divide SYS_CLK by 3	0x2
	CLK_MCLK_PRESCALE_4	Divide SYS_CLK by 4	0x3
	CLK_MCLK_PRESCALE_5	Divide SYS_CLK by 5	0x4
	CLK_MCLK_PRESCALE_6	Divide SYS_CLK by 6	0x5
	CLK_MCLK_PRESCALE_7	Divide SYS_CLK by 7	0x6
	CLK_MCLK_PRESCALE_8	Divide SYS_CLK by 8	0x7

1. Only division factors up to 8 are specified. Both CLKDIV_CFG_PCLK_PRESCALE and CLKDIV_CFG_MCLK_PRESCALE are 7-bit fields; thus division factors up to 128 are possible for both clocks. Division factors above 8 do not have an associated #define statement.

9.3.4.6 D_USRCLKDIV_CFG Settings

Bit Field	Field Name	Field Description
11:0	USRCLKDIV_CFG_USRCLK_PRESCALE	Set prescale factor used to generate configurable USRCLK

Field Name	Value Symbol	Value Description	Hex Value
USRCLKDIV_CFG_USRCLK_PRESCALE ¹	CLK_USRCLK_PRESCALE_1*	Divide SYS_CLK by 1	0x0
	CLK_USRCLK_PRESCALE_2	Divide SYS_CLK by 2	0x1
	CLK_USRCLK_PRESCALE_3	Divide SYS_CLK by 3	0x2
	CLK_USRCLK_PRESCALE_4	Divide SYS_CLK by 4	0x3
	CLK_USRCLK_PRESCALE_5	Divide SYS_CLK by 5	0x4
	CLK_USRCLK_PRESCALE_6	Divide SYS_CLK by 6	0x5
	CLK_USRCLK_PRESCALE_7	Divide SYS_CLK by 7	0x6
	CLK_USRCLK_PRESCALE_8	Divide SYS_CLK by 8	0x7
	CLK_USRCLK_PRESCALE_9	Divide SYS_CLK by 9	0x8
	CLK_USRCLK_PRESCALE_10	Divide SYS_CLK by 10	0x9

1. Only division factors up to 10 are specified. USRCLKDIV_CFG_USRCLK_PRESCALE is a 12-bit field, thus division factors up to 4096 are possible for the clock. Division factors above 10 do not have an associated #define statement.

9.3.4.7 D_WOLADIV_CFG Settings

Bit Field	Field Name	Field Description
11:0	WOLADIV_CFG_WOLACLK_PRESCALE	Set prescale factor used to generate WOLACLK

Field Name	Value Symbol	Value Description	Hex Value
WOLADIV_CFG_WOLACLK_PRESCALE ¹	CLK_WOLACLK_PRESCALE_1_000*	Divide SYS_CLK by 1.000, execution time (1+0/8)T	0x0
	CLK_WOLACLK_PRESCALE_1_125	Divide SYS_CLK by 1.125, execution time (1+1/8)T	0x1
	CLK_WOLACLK_PRESCALE_1_250	Divide SYS_CLK by 1.250, execution time (1+2/8)T	0x2
	CLK_WOLACLK_PRESCALE_1_375	Divide SYS_CLK by 1.375, execution time (1+3/8)T	0x3
	CLK_WOLACLK_PRESCALE_1_500	Divide SYS_CLK by 1.500, execution time (1+4/8)T	0x4
	CLK_WOLACLK_PRESCALE_1_625	Divide SYS_CLK by 1.625, execution time (1+5/8)T	0x5
	CLK_WOLACLK_PRESCALE_1_750	Divide SYS_CLK by 1.750, execution time (1+6/8)T	0x6
	CLK_WOLACLK_PRESCALE_1_875	Divide SYS_CLK by 1.875, execution time (1+7/8)T	0x7
	CLK_WOLACLK_PRESCALE_2_000	Divide SYS_CLK by 2.000, execution time (1+8/8)T	0x8
	CLK_WOLACLK_PRESCALE_3_000	Divide SYS_CLK by 3.000, execution time (1+16/8)T	0x10
	CLK_WOLACLK_PRESCALE_4_000	Divide SYS_CLK by 4.000, execution time (1+24/8)T	0x18
	CLK_WOLACLK_PRESCALE_8_000	Divide SYS_CLK by 8.000, execution time (1+56/8)T	0x38
	CLK_WOLACLK_PRESCALE_16_000	Divide SYS_CLK by 16.000, execution time (1+120/8)T	0x78
	CLK_WOLACLK_PRESCALE_24_000	Divide SYS_CLK by 24.000, execution time (1+184/8)T	0xB8
	CLK_WOLACLK_PRESCALE_32_000	Divide SYS_CLK by 32.000, execution time (1+248/8)T	0xF8
	CLK_WOLACLK_PRESCALE_64_000	Divide SYS_CLK by 64.000, execution time (1+504/8)T	0x1F8
	CLK_WOLACLK_PRESCALE_128_000	Divide SYS_CLK by 128.000, execution time (1+1016/8)T	0x3F8

1. These are some common factors included as #define statements for convenience. Any division factor up to 128.875 can be used but may not have a corresponding symbol.

9.3.4.8 D_STANDBY_CFG Settings

Bit Field	Field Name	Field Description
1	STANDBY_CFG_CLK	Enable/disable the standby clock

Field Name	Value Symbol	Value Description	Hex Value
STANDBY_CFG_CLK	STANDBY_CLK_DISABLE*	Disable the standby clock	0x0
	STANDBY_CLK_ENABLE	Enable the standby clock	0x1

9.4 GENERAL-PURPOSE TIMER

Orela 4500 has a 12-bit general-purpose timer with a 3-bit prescaler that can be configured to run in either single-shot or continuous mode. When the timer expires, it generates a TIMER interrupt. In single-shot mode, one interrupt is generated and the timer waits to be started again; in free-run mode, the timer will reset, start again automatically, and generate interrupts every time it expires until it is stopped.

To reset the timer, the timer will have to be configured for single-shot mode and the timer value (TIMER_CFG_DELAY) will have to be set to one (the smallest possible value). This will cause the timer to stop one PCLK cycle after the timer value has been configured, provided that the timer has already been started. The timer is operating in the background and as such it is not possible to monitor the status of the timer on an ongoing basis.

9.4.1 Configuration and Operation

The D_TIMER_CFG configuration register controls the mode of the timer. The 12-bit value TIMER_CFG_DELAY sets a delay of (n+1) timer clock cycles. The TIMER_CFG_PRESCALE bit field sets the divisor that is applied to the peripheral clock (PCLK) to generate the timer clock.

The TIMER_CFG_MODE bit sets the timer for either free-run mode or single-shot mode.

9.4.2 Starting the Timer

Setting the SYS_CTRL_TIMER_START bit in the D_SYS_CTRL register starts the timer. As long as the timer is running, a read from this bit as SYS_CTRL_TIMER_BUSY will indicate 1.

9.4.3 Stopping the Timer

In single-shot mode, the timer stops after generating a single interrupt. To stop the timer in free-run mode, set the timer to single-shot mode. The timer stops after the next interrupt.

9.4.4 Control and Configuration Registers

Register Name	Register Description	Address
D_SYS_CTRL	System control register used to start the general purpose timer	EXT7 (RCore register)
D_TIMER_CFG	General Purpose Timer configuration	Y:0x8005

9.4.4.1 D_SYS_CTRL Settings

Bit Field	Field Name	Field Description
3	SYS_CTRL_TIMER_START	Starts the timer

9.4.4.2 D_TIMER_CFG Settings

Bit Field	Field Name	Field Description
15	TIMER_CFG_MODE	Timer configuration for one-shot or free-run mode
14:12	TIMER_CFG_PRESCALE	Number of peripheral clock cycles per timer clock cycle
11:0	TIMER_CFG_DELAY	Timer expires after (Timer Delay + 1) timer clock cycles

Field Name	Value Symbol	Value Description	Hex Value
TIMER_CFG_MODE	TIMER_ONESHOT*	Select single-shot mode	0x0
	TIMER_FREERUN	Select free-run mode	0x1
TIMER_CFG_PRESCALE	TIMER_PRESCALE_1*	Divide PCLK by 1	0x0
	TIMER_PRESCALE_2	Divide PCLK by 2	0x1
	TIMER_PRESCALE_4	Divide PCLK by 4	0x2
	TIMER_PRESCALE_8	Divide PCLK by 8	0x3
	TIMER_PRESCALE_16	Divide PCLK by 16	0x4
	TIMER_PRESCALE_32	Divide PCLK by 32	0x5
	TIMER_PRESCALE_64	Divide PCLK by 64	0x6
	TIMER_PRESCALE_128	Divide PCLK by 128	0x7

9.5 MULTI-CHIP SYNCHRONIZATION

Multi-chip synchronization allows for sample clock synchronization across multiple Orela 4500 chips. “Synchronized sample clocks” is defined as an operating condition where all A/D converters in a configuration of multiple Orela 4500 chips sample incoming signals at the same instants in time. A configuration of multiple Orela 4500 chips is defined as a configuration with interconnected Orela 4500 chips that are all operating from the same system clock source.

For example, sample clock synchronization can be useful in beamforming applications that involve microphone arrays with more than two microphones (one Orela 4500 has a maximum of two simultaneous inputs) where sample phase and timing correlation are critical.

A control register bit field, GPIO_PIN_CFG_CLKRST_EBL in the D_GPIO_PIN_CFG control register is used to enable an input signal to Orela 4500 that is used to synchronize MCLK between the transmitter of the synchronization signal (master) and the receiver of the synchronization signal (slave). Multiple slaves are allowed. When synchronization is performed, the MCLK clock signal is brought in phase across chips. Having MCLK in the same phase across chips is necessary to obtain samples at the same moments in time because the A/D converters utilize this clock to sample the incoming signals. The synchronization input on the slave takes place on the GPIO[3] pin. The synchronization signal delivered by the master is delivered on the master's GPIO[3] pin.

Note: For multi-chip synchronization to be performed, it is essential that all chips in the configuration run at the same SYS_CLK. SYS_CLK can be delivered by one of the chips in the multi-chip configuration or it can be delivered to all chips by an external source.

The GPIO[3] pin must be set for the correct direction in the D_GPIO_DIR_CFG control register (output for master, input for slave).

Enabling synchronization in the GPIO_PIN_CFG_CLKRST_EBL bit field in the D_GPIO_PIN_CFG control register on the slave ties GPIO[3] to the subcircuit that generates MCLK from SYS_CLK. A single transition from low to high on the GPIO[3] input pin on the slave will then result in the slave obtaining in-phase MCLK with the master. The master delivers the single transition on its GPIO[3] output pin.

For more information about how to configure the `GPIO_PIN_CFG_CLKRST_EBL` bit field in the `D_GPIO_PIN_CFG` control register, see Section 8.1, “General Purpose I/O (GPIO)” on page 69.

An example of how to connect one master to multiple slaves in a configuration of multiple Orela 4500 chips is shown in Figure 32. Here, the master must bring itself and all slaves into synchronized sample operation. In this configuration the master delivers the system clock on the master `EXT_CLK` pin (configured as output); all slaves receive a system clock on the slave `EXT_CLK` pins (configured as inputs). The master delivers a synchronization signal on the `GPIO[3]` pin; the slaves receive the synchronization signal on the `GPIO[3]` pins. For information about configuring `EXT_CLK` see Section 9.3, “Oscillator Circuitry and Clock Generation” on page 107.

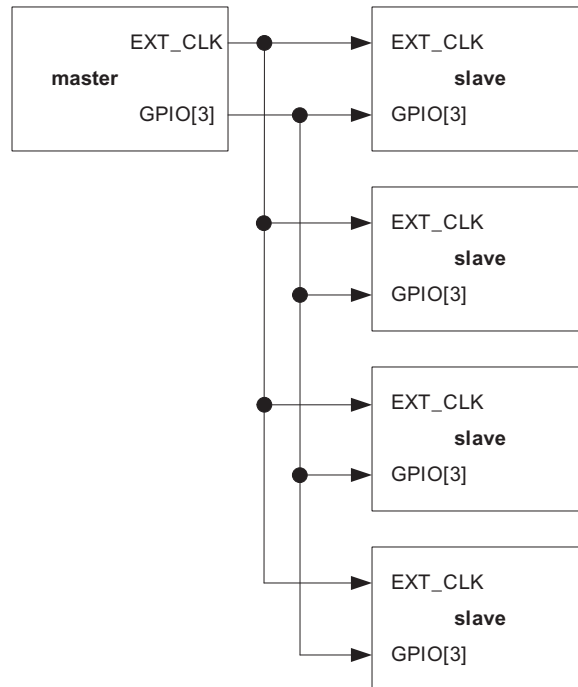


Figure 32: Synchronization of multiple Orela 4500 chips.

9.6 POWER-ON RESET

During the power-on procedure, all audio outputs are muted and all Orela 4500 configuration and control registers (both analog and digital) are set to their default values as specified throughout the rest of this reference manual. RCore data registers, address registers, and control registers are set to zero. The contents of RAM are unspecified.

Approximately 220 ms elapse between the device first powering on and the first line of code being loaded from the EEPROM. This delay is due to the approximately 20 ms of the power-on reset timer, after which execution starts at address `P:0x0000` in the Boot ROM and a 200 ms delay occurs, comprising:

- 50 ms delay for the voltage supply switch and stabilization

- 150 ms for Boot ROM initialization

For more details see the Boot ROM section of the *Orela 4500 Firmware Reference Manual*.

9.7 POWER SUPPLY REGULATION AND MANAGEMENT

The power supply is a critical component for overall system performance (especially audio performance).

The power supply provides:

- A regulated, low-noise power source for all analog sub-systems on Orela 4500
- A regulated, noise-robust voltage for microphones
- A configurable regulated supply for the RCore, WOLA, IOP, interfaces, and peripherals
- A voltage translation scheme for the SPI interface to allow for run-time data-logging (see Section 8.6, “SPI Port” on page 86 for more details)
- An on-chip charge pump
- A soft power-down mode that puts all on-chip subsystems into an ultra-low power consumption sleep mode
- A built-in power management unit that is used to ensure safe system operation

Three basic power-supply modes exist: Low Voltage (LV), High Voltage (HV), and Double Voltage (DV). The supply mode is set in the A_PSU_CTRL control register using the PSU_CTRL_VDDC_MODE bit field. The three modes are described in Table 15. The default power supply mode depends on the signal at GPIO[15] at power-on. If GPIO[15] is asserted, Orela 4500 will boot in LV mode; otherwise, it will boot in HV mode.

TABLE 15: POWER SUPPLY MODES

Mode	Description
Low-Voltage (LV) (default)	Orela 4500 operates from a nominal supply of 1.25 V. The internal charge pump is enabled and used to supply analog on-chip sub-systems (input, output stages) with a 2 V regulated voltage. The RCore, WOLA, IOP and the interfaces are supplied from an internally generated 1 V regulated supply.
High-Voltage (HV)	Orela 4500 operates from a nominal supply of 1.8 V. The internal charge pump is enabled and supplies the analog on-chip sub-systems (input, output stages) with a 2 V regulated voltage. The RCore, WOLA, IOP and the interfaces are supplied directly from the 1.8 V supply voltage.
Double-Voltage (DV)	Orela 4500 operates from a nominal supply of 1.25 V. The internal charge pump is enabled and used to supply analog on-chip sub-systems (input, output stages) with a 2 V regulated voltage. The RCore, WOLA, IOP and the interfaces are supplied from an internally generated 2 V regulated supply.

When the voltage over the charge pump capacitor is refreshed, the refresh frequency must be an approximate integer multiple of the sample frequency. If the refresh frequency is configured in this way, possible tones generated on the power supply are suppressed in the decimation filters, as they will occur out of the passband of the filter. Also, by having the option of changing the

refresh frequency, the charge pump current can be tuned to match a given load (as a higher load will require a higher charge pump refresh frequency). In a standard configuration using a 100 nF charge pump capacitor between the CAP0 and CAP1 pins and the charge pump configured for a 128 kHz refresh signal frequency, the load regulation is around 120 mV/mA. A higher refresh frequency may be used to improve this voltage regulation level (the load regulation ranges from approximately 100 mV/mA for a refresh rate of 640 kHz to approximately 160 mV/mA for a refresh rate of 80 kHz). The refresh signal is created from MCLK through clock division and the division factor is set in the A_DIV_CLK_CTRL control register using the DIV_CLK_CTRL_CHARGE_PUMP_PRESCALE bit field. For more information about configuring MCLK see Section 9.3, “Oscillator Circuitry and Clock Generation” on page 107.

Figure 33 provides a block diagram overview of the power-supply subsystem.

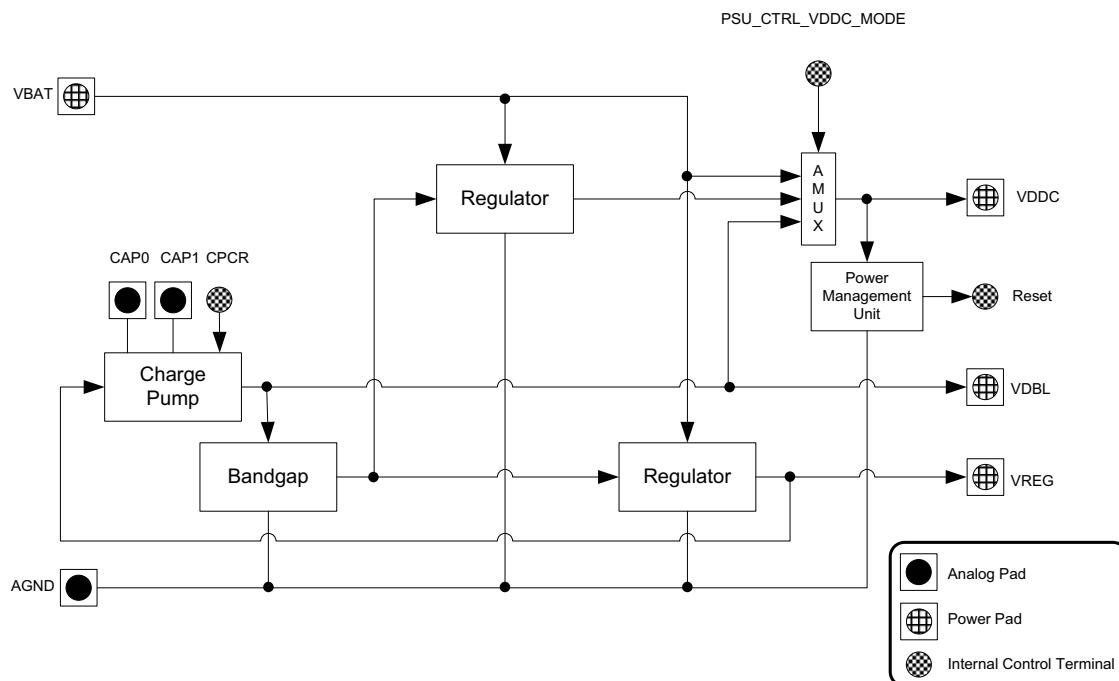


Figure 33: Power Supply Block Diagram.

In Figure 33, VDBL is the doubled voltage, VREG is the regulated voltage for the microphone, VDDC is the regulated supply for the digital I/O pads (and external EEPROM in case this is connected to Orela 4500), and VREGD is an internal power terminal that constitutes the regulated supply for the RCore, WOLA, IOP, and the interfaces.

Note: VDDC must always be connected to VDDO (not shown) as the VDDO input pins constitute the power supply inputs for the digital I/O pads.

9.7.1 Soft Power Down Mode

The Orela 4500 chip provides an ultra-low power mode where the RCore and digital components of the system are still active but are being clocked at approximately 20 kHz and the analog circuitry is disabled. Since the RCore is still active, although running at only 20 kHz, an event on the GPIO

or other system element can be used to trigger a reactivation of the system from soft power down mode.

As the digital interfaces will still be operative in soft power down mode, it is recommended that system elements such as the IOP be disabled in soft power down mode to prevent unexpected behaviour upon restoration of the normal clock. It should be further noted that many interfaces, including specifically the TWSS and UART interfaces, will remain active unless explicitly disabled although they will not be able to operate at typical baud rates as their clocks will be scaled down proportionally with SYS_CLK.

Perform the following sequence of tasks to enter soft power down mode:

1. Enable the standby clock. For more information on enabling the standby clock, see Section 9.3, “Oscillator Circuitry and Clock Generation” on page 107.
2. Switch the source for SYS_CLK to the standby clock. For more information on changing the source of SYS_CLK, see Section 9.3, “Oscillator Circuitry and Clock Generation” on page 107.
3. Disable the analog blocks from the input and output stages by setting the STANDBY_CFG_ANALOG_DISABLE bit in the D_STANDBY_CFG control register.

Note: This ordering is important as the main oscillator is disabled along with the rest of the analog circuitry and cannot be disabled unless the system is operating using the standby clock. For similar reasons, when re-enabling the system in response to an event, the analog circuitry must be enabled before the normal SYS_CLK is restored.

9.7.2 Power Management Unit

The power management unit (PMU) ensures safe system operation under any battery condition, as it will reset the system when the VDDC voltage drops below a certain threshold voltage. This threshold voltage is set to be a safe margin above the voltage at which correct execution of RCore instructions may no longer occur. It is hard-coded into the system and cannot be configured manually. When reset occurs, the power-on reset sequence is performed; see Section 9.6, “Power-On Reset” on page 120. The relevant voltage thresholds are found in Table 16.

TABLE 16: POWER MANAGEMENT THRESHOLDS

Threshold	Voltage Level
VDDC Shutdown	0.83 V $\pm$ 50 mV
VDDC Startup	0.85 V $\pm$ 50 mV

At the VDDC startup voltage threshold, the system will start up during power on or following a system reset due to a voltage drop below the VDDC shutdown voltage threshold. If the system VDDC voltage falls below the shutdown voltage threshold (VDDC Shutdown), even if the voltage drop is due to a temporary transient, the system will shut down and not restart until the VDDC startup voltage is reached. The voltage difference between VDDC startup and VDDC shutdown will prevent repeated system resets due to voltage oscillation around the VDDC shutdown voltage.

9.7.3 Control and Configuration Registers

Register Name	Register Description	Address
A_DIV_CLK_CTRL	Configure division factor	A:0x13
A_PSU_CTRL	Configure power supply mode	A:0x1A
D_STANDBY_CFG	Analog circuitry disable for soft power down	Y:0x8011

9.7.3.1 A_DIV_CLK_CTRL Settings

Bit Field	Field Name	Field Description
3:0	DIV_CLK_CTRL_CHARGEUMP_PRESALE	Set division factor for charge pump refresh frequency generation

Field Name	Value Symbol	Value Description	Hex Value
DIV_CLK_CTRL_CHARGEUMP_PRESALE	CHARGEUMP_DISABLE	Charge pump off	0x0
	CHARGEUMP_PRESALE_2	Use MCLK/2	0x1
	CHARGEUMP_PRESALE_3	Use MCLK/3	0x2
	CHARGEUMP_PRESALE_4	Use MCLK/4	0x3
	CHARGEUMP_PRESALE_5	Use MCLK/5	0x4
	CHARGEUMP_PRESALE_6	Use MCLK/6	0x5
	CHARGEUMP_PRESALE_7	Use MCLK/7	0x6
	CHARGEUMP_PRESALE_8	Use MCLK/8	0x7
	CHARGEUMP_PRESALE_9	Use MCLK/9	0x8
	CHARGEUMP_PRESALE_10*	Use MCLK/10	0x9
	CHARGEUMP_PRESALE_11	Use MCLK/11	0xA
	CHARGEUMP_PRESALE_12	Use MCLK/12	0xB
	CHARGEUMP_PRESALE_13	Use MCLK/13	0xC
	CHARGEUMP_PRESALE_14	Use MCLK/14	0xD
	CHARGEUMP_PRESALE_15	Use MCLK/15	0xE
	CHARGEUMP_PRESALE_16	Use MCLK/16	0xF

9.7.3.2 A_PSU_CTRL Settings

Bit Field	Field Name	Field Description
1:0	PSU_CTRL_VDDC_MODE	Select power supply mode

Field Name	Value Symbol	Value Description	Hex Value
PSU_CTRL_VDDC_MODE	VDDC_MODE_LOW_VOLTAGE*	Select LV mode	0x0
	VDDC_MODE_HIGH_VOLTAGE	Select HV mode	0x1
	VDDC_MODE_DOUBLE_VOLTAGE	Select DV mode	0x2

9.7.3.3 D_STANDBY_CFG Settings

Bit Field	Field Name	Field Description
0	STANDBY_CFG_ANALOG_DISABLE	Disable/enable the analog circuit blocks

Field Name	Value Symbol	Value Description	Hex Value
STANDBY_CFG_ANALOG_DISABLE	STANDBY_ANALOGBLOCK_ENABLE*	Enable analog blocks	0x0
	STANDBY_ANALOGBLOCK_DISABLE	Disable analog blocks	0x1

9.8 WATCHDOG TIMER

The watchdog timer is a simple safety system that in most cases will reset a system that has malfunctioned. It is a countdown timer that must be periodically acknowledged before it reaches zero. The system assumes that the application's failure to acknowledge this countdown timer before it reaches zero is an indication that the system is malfunctioning and must be reset. The countdown timer value for the watchdog timer is not visible to the RCore. The watchdog timer cannot be disabled.

The watchdog is a 12-bit counter that is clocked by PCLK. For most applications PCLK runs at 1.28 MHz. For other values of PCLK, time values given in the configuration settings outlined in Section 9.8.2, "Control and Configuration Registers" on page 126 must be scaled accordingly. For more information about configuring PCLK see Section 9.3, "Oscillator Circuitry and Clock Generation" on page 107.

The watchdog timeout may be configured to a range of time delays from 1.6 milliseconds to just over 3.2 seconds using the WATCHDOG_CFG_TIMEOUT bit field in the D_WATCHDOG_CFG register, assuming an MCLK of 1.28 MHz.

9.8.1 Operation

Code running on Orela 4500 should acknowledge the watchdog timer before it expires to avoid a system reset. To acknowledge the watchdog timer, the SYS_CTRL_WATCHDOG_RESET bit field is set to 1 in the D_SYS_CTRL register.

When the watchdog timer expires for the first time, it generates a "warning" interrupt. The interrupt service routine should not normally be used to acknowledge the watchdog timer, since

interrupts may still work correctly even when the main application has crashed. The interrupt service routine should be used to do a soft reset and set the system into an alarm state. An alarm state means that audio muting and other necessary tasks are performed followed by an endless wait loop that is executed until the watchdog expires a second time.

If the watchdog is allowed to expire a second time without acknowledgement, it is assumed that the system has entered an invalid state and a system reset occurs. The power-on reset sequence is performed (see Section 9.6, "Power-On Reset" on page 120 for details); however, since the system does not lose power, the contents of RAM are preserved except as modified by the boot process.

9.8.2 Control and Configuration Registers

Register Name	Register Description	Address
D_SYS_CTRL	System Control Register used to reset the watchdog	EXT7 (RCore register)
D_WATCHDOG_CFG	Watchdog timer configuration	Y:0x800A

9.8.2.1 D_SYS_CTRL Settings

Bit Field	Field Name	Field Description
4	SYS_CTRL_WATCHDOG_RESET	Acknowledge the watchdog timer

Field Name	Value Symbol	Value Description	Hex Value
SYS_CTRL_WATCHDOG_RESET	SYS_WATCHDOG_RESET	Acknowledge the watchdog timer	0x1

9.8.2.2 D_WATCHDOG_CFG Settings

Bit Field	Field Name	Field Description
3:0	WATCHDOG_CFG_TIMEOUT	Timeout for watchdog reset (assumes 1.28 MHz PCLK)

Field Name	Value Symbol	Value Description	Hex Value
WATCHDOG_CFG_TIMEOUT	WATCHDOG_TIMEOUT_1M6	1.6 ms timeout	0x0
	WATCHDOG_TIMEOUT_3M2	3.2 ms timeout	0x1
	WATCHDOG_TIMEOUT_6M4	6.4 ms timeout	0x2
	WATCHDOG_TIMEOUT_12M8	12.8 ms timeout	0x3
	WATCHDOG_TIMEOUT_25M6	25.6 ms timeout	0x4
	WATCHDOG_TIMEOUT_51M2	51.2 ms timeout	0x5
	WATCHDOG_TIMEOUT_102M4	102.4 ms timeout	0x6
	WATCHDOG_TIMEOUT_204M8	204.8 ms timeout	0x7
	WATCHDOG_TIMEOUT_409M6	409.6 ms timeout	0x8
	WATCHDOG_TIMEOUT_819M2	819.2 ms timeout	0x9
	WATCHDOG_TIMEOUT_1638M4	1638.4 ms timeout	0xA
	WATCHDOG_TIMEOUT_3276M8	3276.8 ms timeout	0xB

9.9 CONFIGURATION REGISTER CONTROLLER

The Orela 4500 chip provides a register setting which prevents the application from writing to all registers in the Y:0x8000 to Y:0x8012 range. This may be used to protect these registers against changes in configuration, which may change system behaviour from a baseline setting. When the CFG_REG_LOCK bit of the D_CFG_REG_CTRL register is used to enable this feature and lock the registers, writes to the locked registers fail silently allowing the written register to maintain the currently held value.

9.9.1 Control and Configuration Registers

Register Name	Register Description	Address
D_CFG_REG_CTRL	Configuration register control register	Y:0x4005

9.9.1.1 D_CFG_REG_CTRL Settings

Bit Field	Field Name	Field Description
0	CFG_REG_LOCK	Lock/unlock control and configuration registers

Field Name	Value Symbol	Value Description	Hex Value
CFG_REG_LOCK	CFG_REG_UNLOCK*	Registers in the Y:8000 block are unlocked	0x0
	CFG_REG_LOCK	Registers in the Y:8000 block are locked	0x1

Sampling Frequencies

A.1 SAMPLING FREQUENCY OVERVIEW

Table 17 summarizes a number of possible sampling frequencies and the relationship between desired MCLK frequencies and the associated sampling frequencies. All values are given in Hz.

TABLE 17: SAMPLING FREQUENCY SUMMARY

ADC_CTRL_ SAMPLE_FREQ	MCLK=640 kHz SYS_CLK=n*MCLK	MCLK=1.28 MHz SYS_CLK=n*MCLK	MCLK=1.92 MHz SYS_CLK=n*MCLK	MCLK=2.56 MHz SYS_CLK=n*MCLK	MCLK=3.84 MHz SYS_CLK=n*MCLK
0x00	10000.00	20000.00	30000.00	40000.00	60000.00
0x01	8888.89	17777.78	26666.67	35555.56	53333.33
0x02	8000.00	16000.00	24000.00	32000.00	48000.00
0x03	7272.73	14545.45	21818.18	29090.91	43636.36
0x04	6666.67	13333.33	20000.00	26666.67	40000.00
0x05	6153.85	12307.69	18461.54	24615.38	36923.08
0x06	5714.29	11428.57	17142.86	22857.14	34285.71
0x07	5333.33	10666.67	16000.00	21333.33	32000.00
0x08	5000.00	10000.00	15000.00	20000.00	30000.00
0x09	4705.88	9411.76	14117.65	18823.53	28235.29
0x0A	4444.44	8888.89	13333.33	17777.78	26666.67
0x0B	4210.53	8421.05	12631.58	16842.11	25263.16
0x0C	4000.00	8000.00	12000.00	16000.00	24000.00
0x0D	3809.52	7619.05	11428.57	15238.10	22857.14
0x0E	3636.36	7272.73	10909.09	14545.45	21818.18
0x0F	3478.26	6956.52	10434.78	13913.04	20869.57
0x10	3333.33	6666.67	10000.00	13333.33	20000.00
0x11	3200.00	6400.00	9600.00	12800.00	19200.00
0x12	3076.92	6153.85	9230.77	12307.69	18461.54
0x13	2962.96	5925.93	8888.89	11851.85	17777.78
0x14	2857.14	5714.29	8571.43	11428.57	17142.86
0x15	2758.62	5517.24	8275.86	11034.48	16551.72
0x16	2666.67	5333.33	8000.00	10666.67	16000.00
0x17	2580.65	5161.29	7741.94	10322.58	15483.87
0x18	2500.00	5000.00	7500.00	10000.00	15000.00
0x19	2424.24	4848.48	7272.73	9696.97	14545.45
0x1A	2352.94	4705.88	7058.82	9411.76	14117.65

TABLE 17: SAMPLING FREQUENCY SUMMARY (CONTINUED)

ADC_CTRL_ SAMPLE_FREQ	MCLK=640 kHz SYS_CLK=n*MCLK	MCLK=1.28 MHz SYS_CLK=n*MCLK	MCLK=1.92 MHz SYS_CLK=n*MCLK	MCLK=2.56 MHz SYS_CLK=n*MCLK	MCLK=3.84 MHz SYS_CLK=n*MCLK
0x1B	2285.71	4571.43	6857.14	9142.86	13714.29
0x1C	2222.22	4444.44	6666.67	8888.89	13333.33
0x1D	2162.16	4324.32	6486.49	8648.65	12972.97
0x1E	2105.26	4210.53	6315.79	8421.05	12631.58
0x1F	2051.28	4102.56	6153.85	8205.13	12307.69
0x20	2000.00	4000.00	6000.00	8000.00	12000.00
0x21	1951.22	3902.44	5853.66	7804.88	11707.32
0x22	1904.76	3809.52	5714.29	7619.05	11428.57
0x23	1860.47	3720.93	5581.40	7441.86	11162.79
0x24	1818.18	3636.36	5454.55	7272.73	10909.09
0x25	1777.78	3555.56	5333.33	7111.11	10666.67
0x26	1739.13	3478.26	5217.39	6956.52	10434.78
0x27	1702.13	3404.26	5106.38	6808.51	10212.77
0x28	1666.67	3333.33	5000.00	6666.67	10000.00
0x29	1632.65	3265.31	4897.96	6530.61	9795.92
0x2A	1600.00	3200.00	4800.00	6400.00	9600.00
0x2B	1568.63	3137.25	4705.88	6274.51	9411.76
0x2C	1538.46	3076.92	4615.38	6153.85	9230.77
0x2D	1509.43	3018.87	4528.30	6037.74	9056.60
0x2E	1481.48	2962.96	4444.44	5925.93	8888.89
0x2F	1454.55	2909.09	4363.64	5818.18	8727.27
0x30	1428.57	2857.14	4285.71	5714.29	8571.43
0x31	1403.51	2807.02	4210.53	5614.04	8421.05
0x32	1379.31	2758.62	4137.93	5517.24	8275.86
0x33	1355.93	2711.86	4067.80	5423.73	8135.59
0x34	1333.33	2666.67	4000.00	5333.33	8000.00
0x35	1311.48	2622.95	3934.43	5245.90	7868.85
0x36	1290.32	2580.65	3870.97	5161.29	7741.94
> 0x36	1269.84	2539.68	3809.52	5079.37	7619.05

PCM Timing Diagrams

B.1 TIMING DIAGRAMS EXPLANATION

- The following diagrams show the PCM interface behaviour with respect to four PCM configuration settings. The configuration settings and how their related diagrams differ are:
- PCM_SUBFRAME bit, used to select between 32-bit frames and 16-bit subframes
- PCM_BIT_ORDER bit, used to select between least significant bit (LSB) first and most significant bit (MSB) first bit ordering
- PCM_FRAME_ALIGNMENT bit, used to select between frame alignment to the first bit (delayed) or to the last bit of data (advanced) within each frame
- PCM_FRAME_WIDTH bit, used to select between a single-bit duration frame signal (short) and a 50% duty cycle frame signal (wide)

B.2 16-BIT TIMING DIAGRAMS

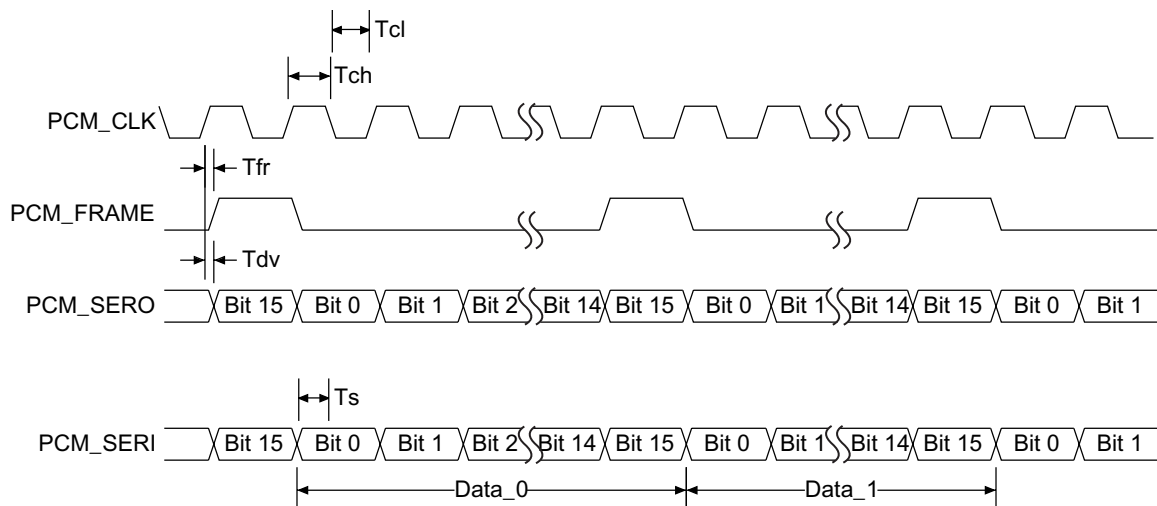


Figure 34: 16-bit, LSB Ordering, Advanced Alignment, Short Frame

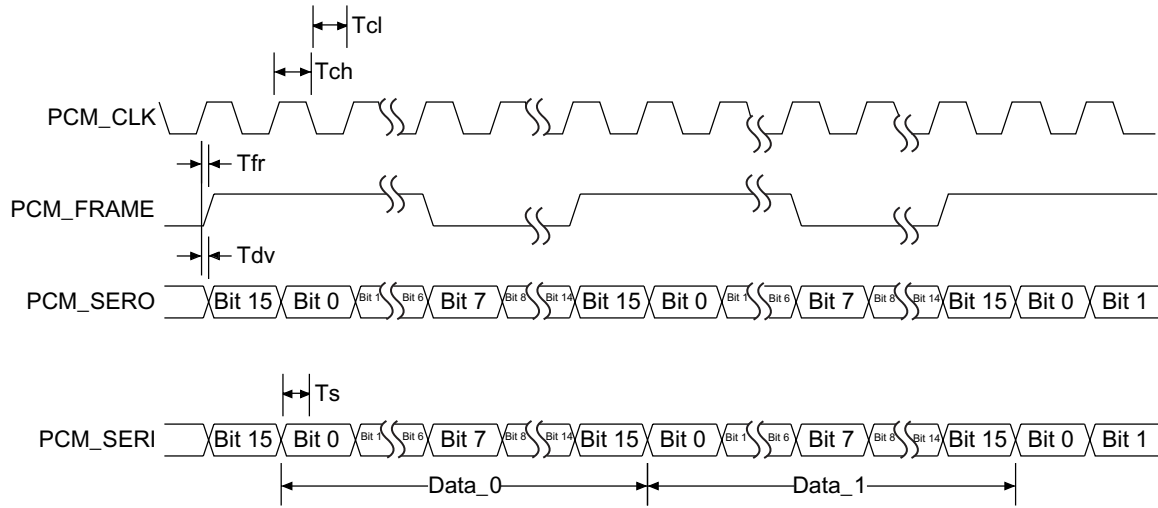


Figure 35: 16-bit, LSB Ordering, Advanced Alignment, Wide Frame

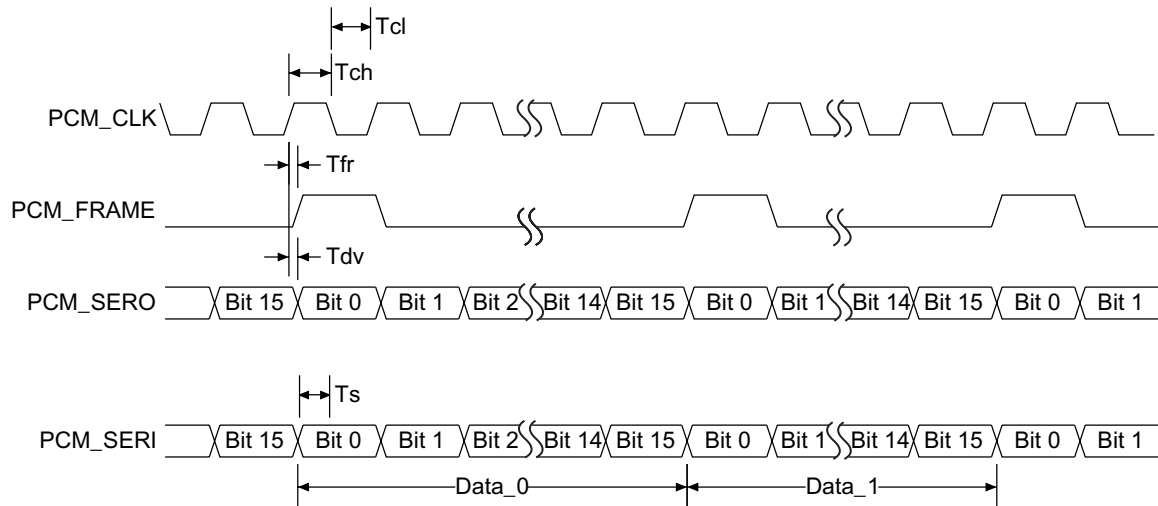


Figure 36: 16-bit, LSB Ordering, Delayed Alignment, Short Frame

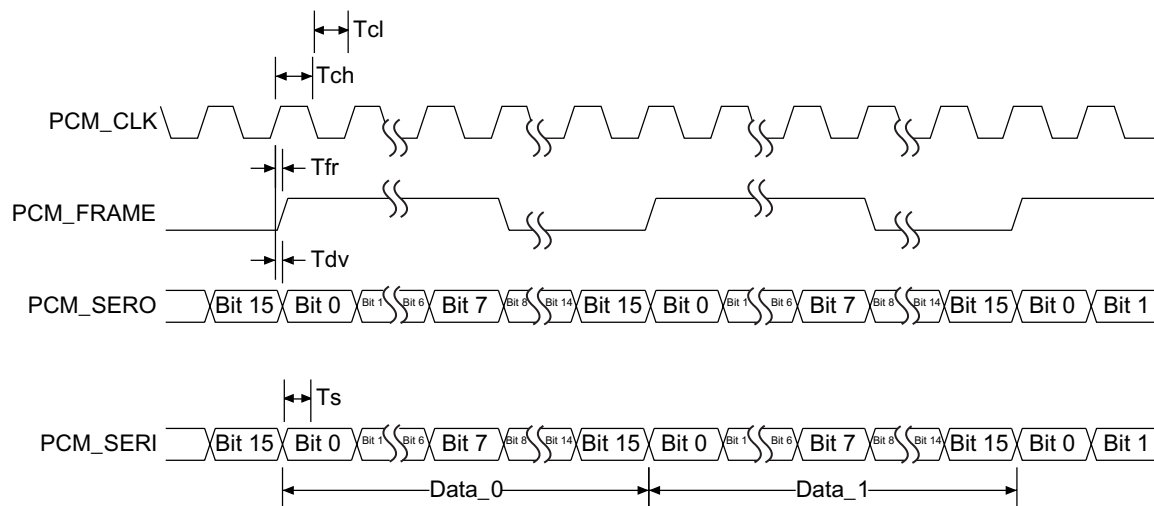


Figure 37: 16-bit, LSB Ordering, Delayed Alignment, Wide Frame

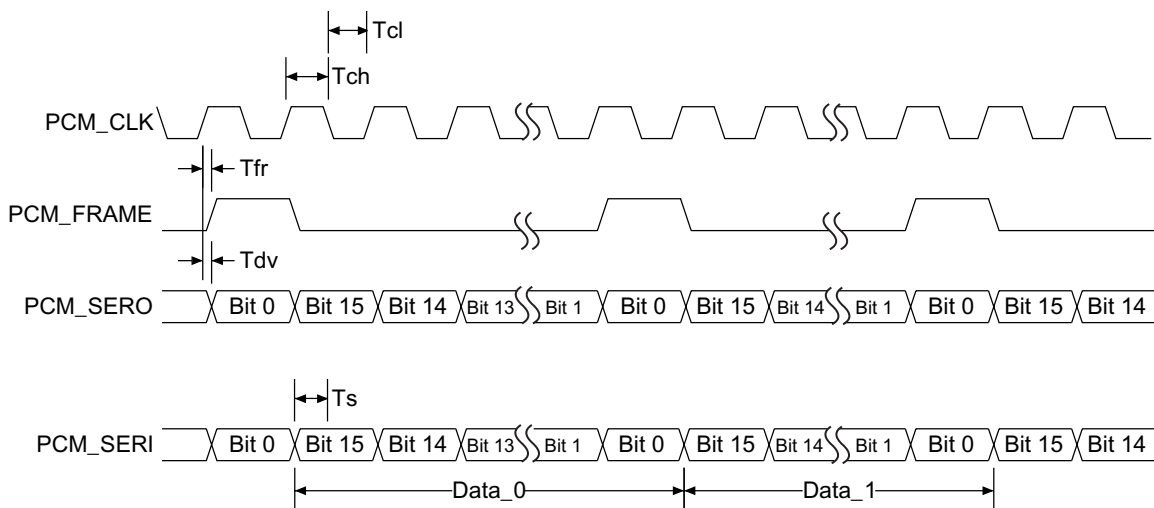


Figure 38: 16-bit, MSB Ordering, Advanced Alignment, Short Frame

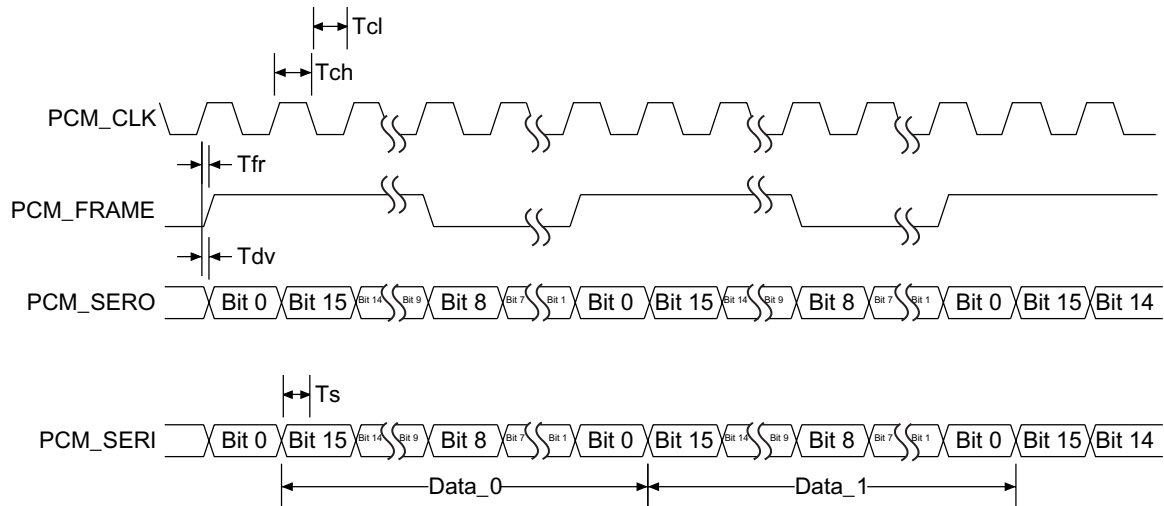


Figure 39: 16-bit, MSB Ordering, Advanced Alignment, Wide Frame

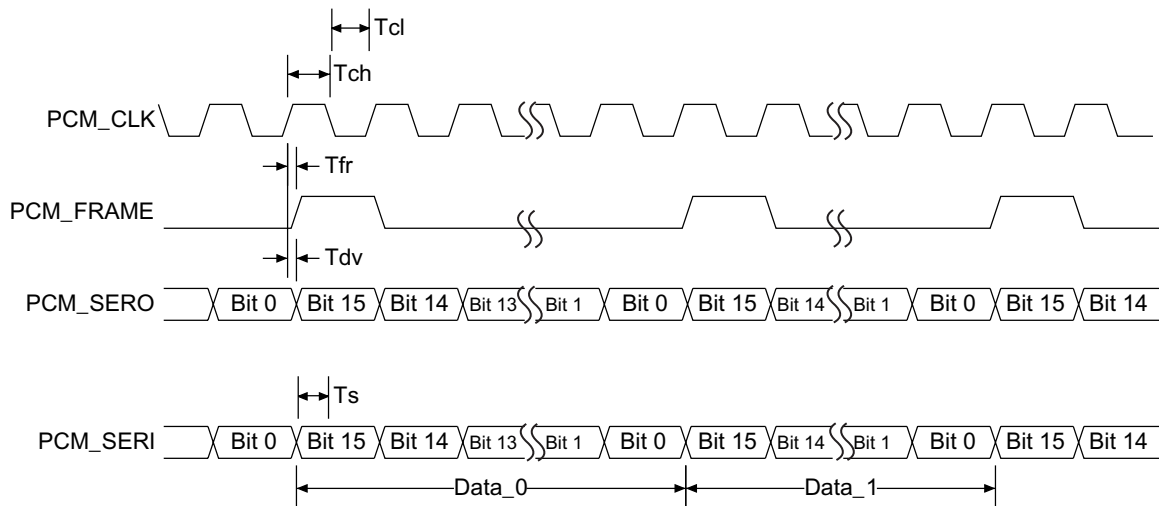


Figure 40: 16-bit, MSB Ordering, Delayed Alignment, Short Frame

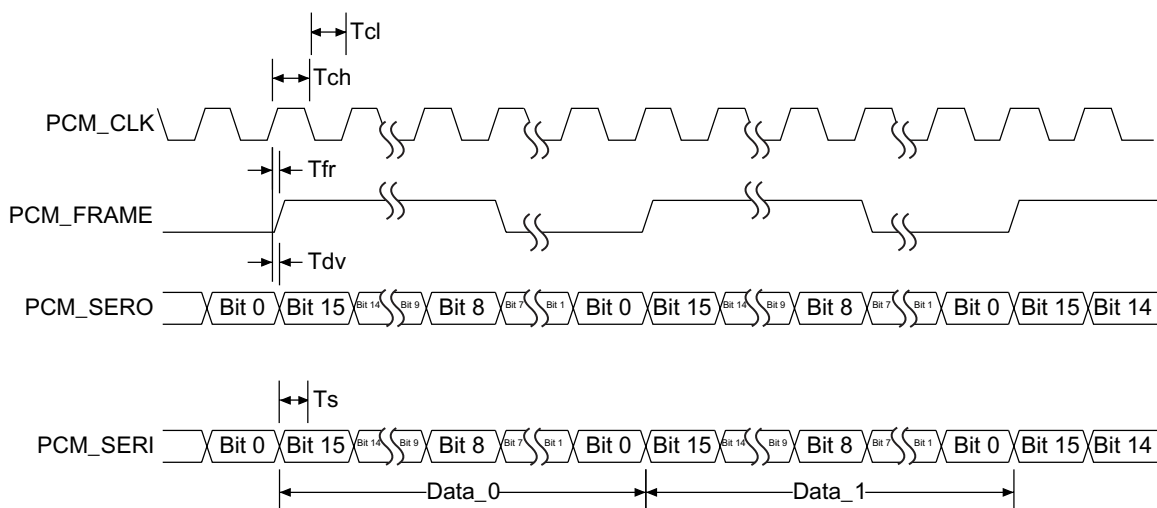


Figure 41: 16-bit, MSB Ordering, Delayed Alignment, Wide Frame

B.3 32-BIT TIMING DIAGRAMS

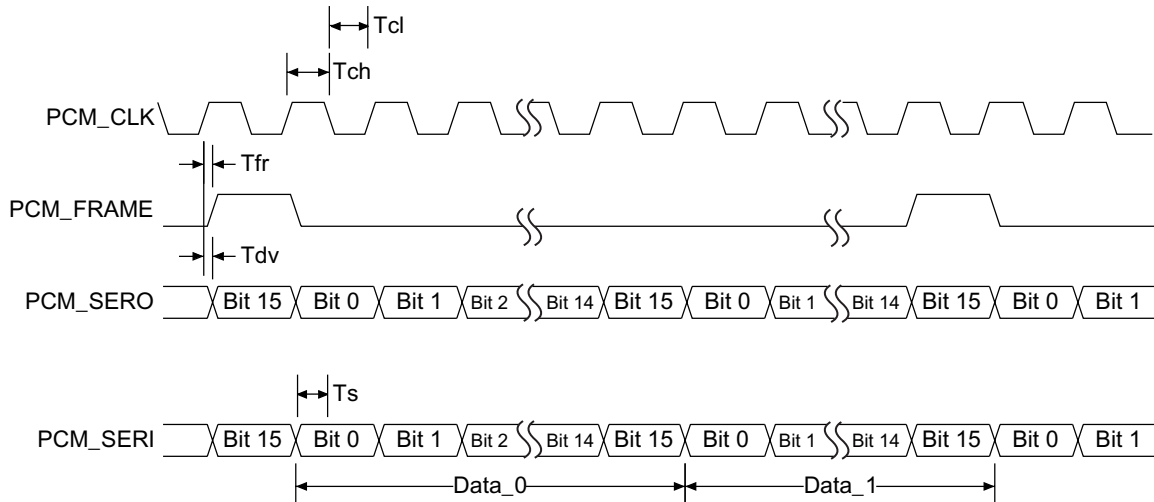


Figure 42: 32-bit, LSB Ordering, Advanced Alignment, Short Frame

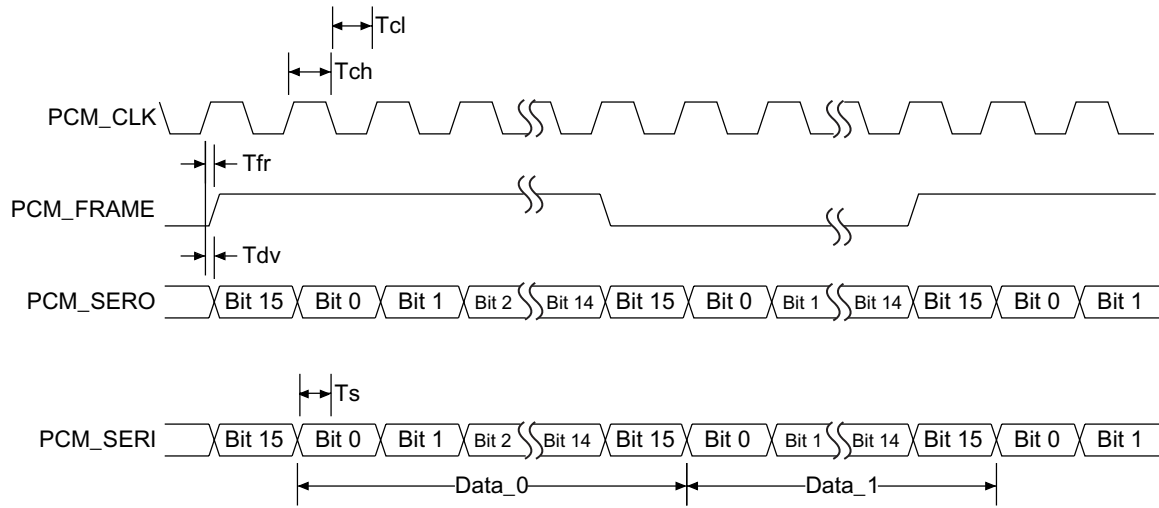


Figure 43: 32-bit, LSB Ordering, Advanced Alignment, Wide Frame

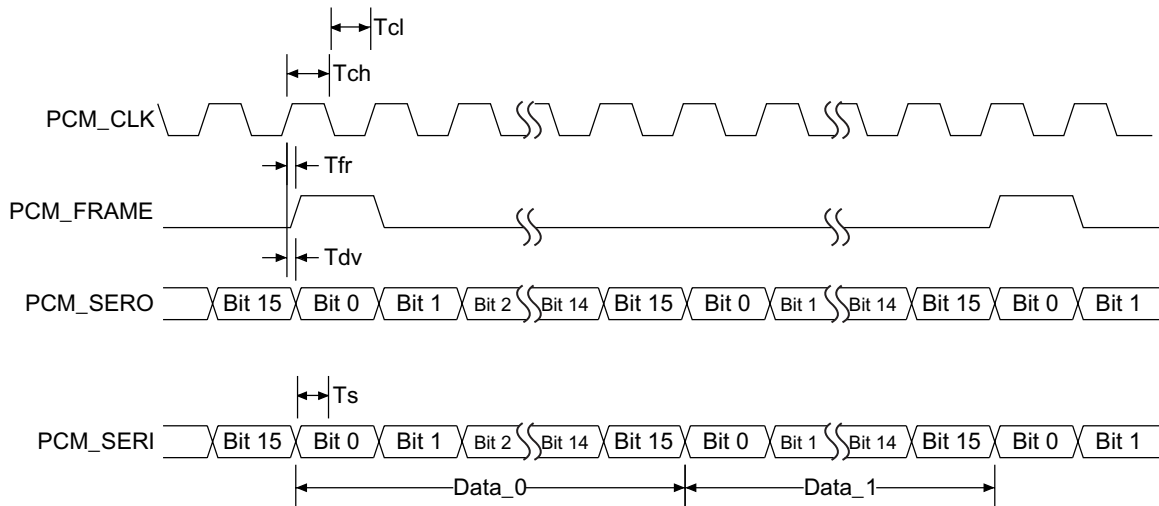


Figure 44: 32-bit, LSB Ordering, Delayed Alignment, Short Frame

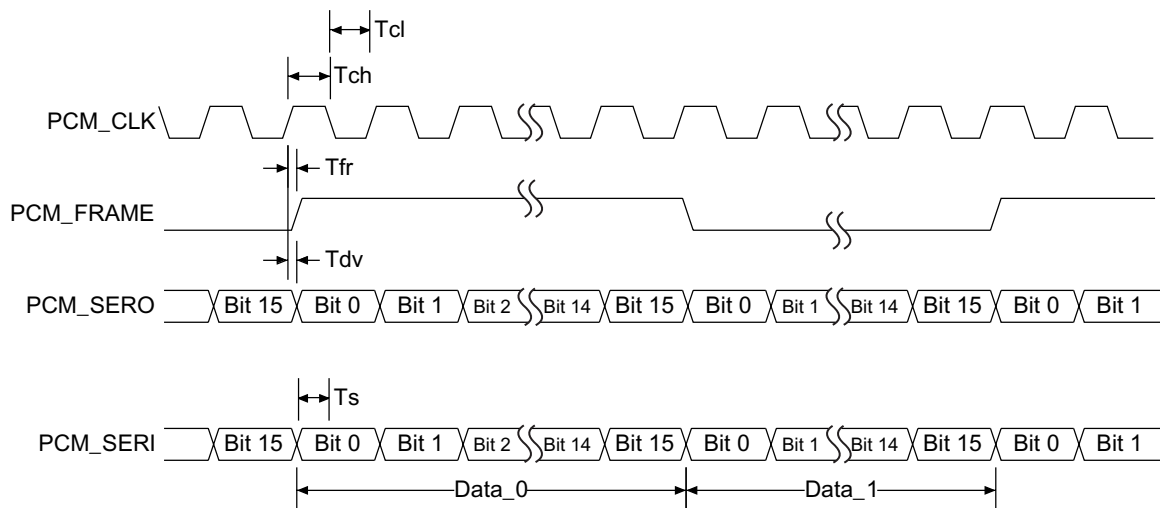


Figure 45: 32-bit, LSB Ordering, Delayed Alignment, Wide Frame

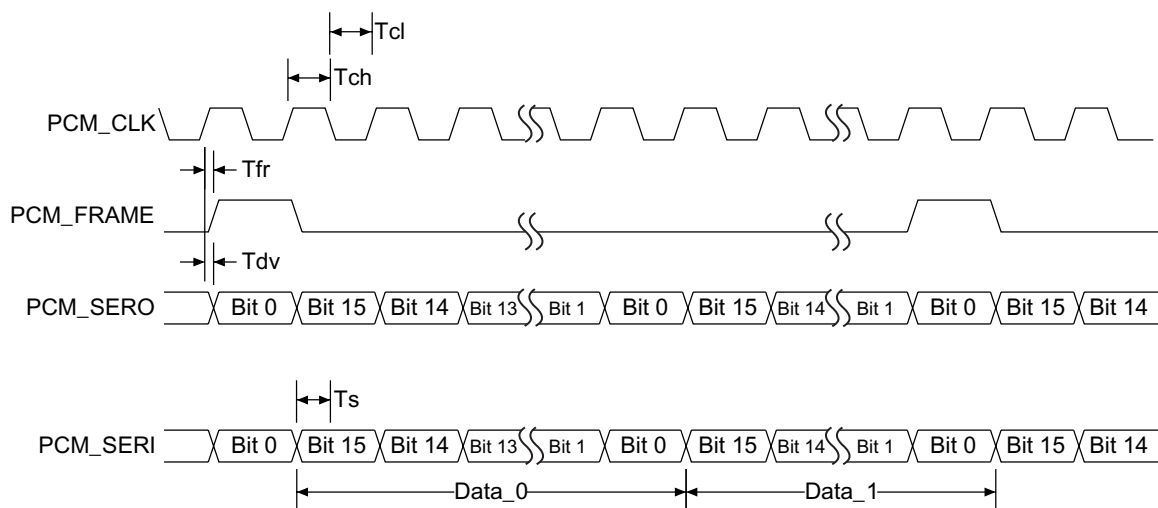


Figure 46: 32-bit, MSB Ordering, Advanced Alignment, Short Frame

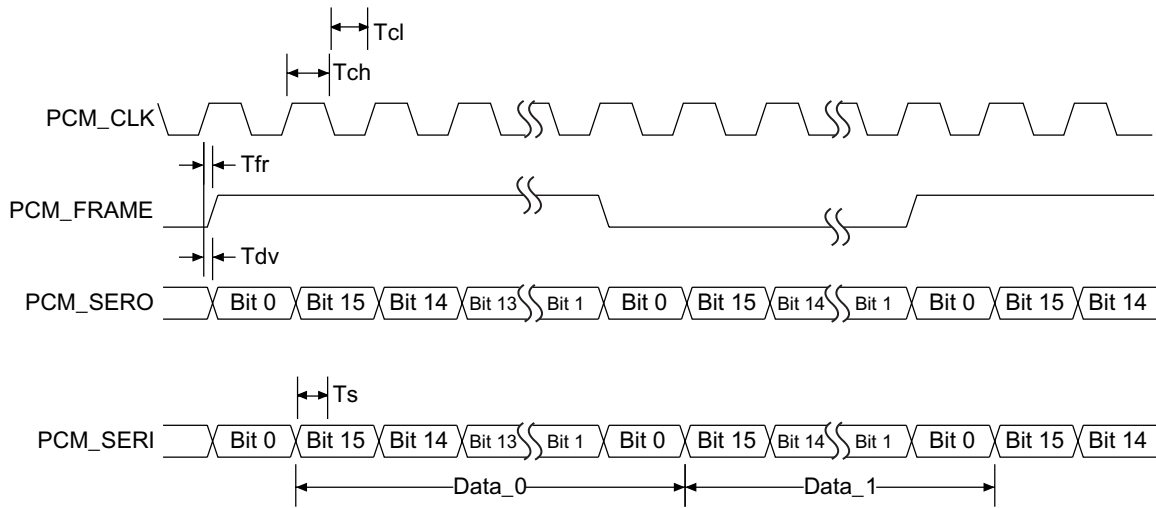


Figure 47: 32-bit, MSB Ordering, Advanced Alignment, Wide Frame

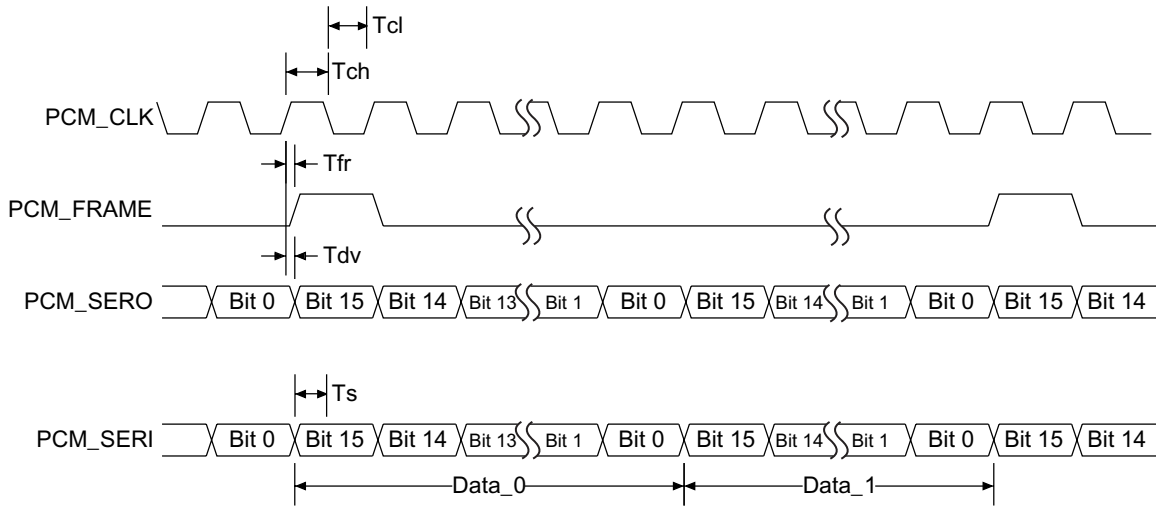


Figure 48: 32-bit, MSB Ordering, Delayed Alignment, Short Frame

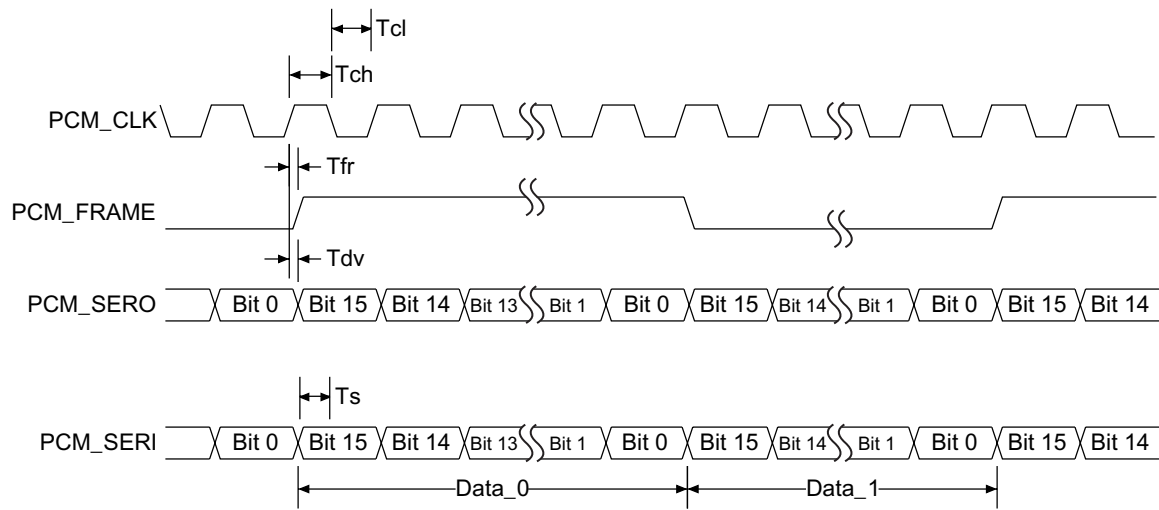


Figure 49: 32-bit, MSB Ordering, Delayed Alignment, Wide Frame

This page left blank for printing purposes.

C

Control and Configuration Register Overview

Refer to the following tables for configuration and control registers:

- Table 18, “Analog Control Registers,” on page 142
- Table 19, “Digital Control Registers,” on page 144
- Table 20, “Configuration Registers,” on page 145
- Table 21, “RCore Digital Control Registers,” on page 147

TABLE 18: ANALOG CONTROL REGISTERS

Address	Register Name	Register Write	Register Read	Default	Description
A:0x00	A_LSAD_CH0_DATA	N/A	(7:0): A_LSAD_CH0_DATA	N/A	LSAD0 channel 0 data
A:0x01	A_LSAD_CH0_CTRL	(4:2): LSAD_CTRL_RANGE N/A	(4:2): LSAD_CTRL_RANGE (1:0): LSAD_CTRL_LSB	0x0 N/A	LSAD0 input dynamic range settings LSBs of LSAD0
A:0x02	A_LSAD_CH1_DATA	N/A	(7:0): A_LSAD_CH1_DATA	N/A	LSAD1 channel 1 data
A:0x03	A_LSAD_CH1_CTRL	(4:2): LSAD_CTRL_RANGE N/A	(4:2): LSAD_CTRL_RANGE (1:0): LSAD_CTRL_LSB	0x0 N/A	LSAD1 input dynamic range settings LSBs of LSAD1
A:0x04	A_LSAD_CH2_DATA	N/A	(7:0): A_LSAD_CH2_DATA	N/A	LSAD2 channel 2 data
A:0x05	A_LSAD_CH2_CTRL	(4:2): LSAD_CTRL_RANGE N/A	(4:2): LSAD_CTRL_RANGE (1:0): LSAD_CTRL_LSB	0x0 N/A	LSAD2 input dynamic range settings LSBs of LSAD2
A:0x06	A_LSAD_CH3_DATA	N/A	(7:0): A_LSAD_CH3_DATA	N/A	LSAD3 channel 3 data
A:0x07	A_LSAD_CH3_CTRL	(4:2): LSAD_CTRL_RANGE N/A	(4:2): LSAD_CTRL_RANGE (1:0): LSAD_CTRL_LSB	0x0 N/A	LSAD3 input dynamic range settings LSBs of LSAD3
A:0x08	A_LSAD_CH4_DATA	N/A	(7:0): A_LSAD_CH4_DATA	N/A	LSAD4 channel 4 data
A:0x09	A_LSAD_CH4_CTRL	(4:2): LSAD_CTRL_RANGE N/A	(4:2): LSAD_CTRL_RANGE (1:0): LSAD_CTRL_LSB	0x0 N/A	LSAD4 input dynamic range settings LSBs of LSAD4
A:0x0A	A_LSAD_CH5_DATA	N/A	(7:0): A_LSAD_CH5_DATA	N/A	LSAD5 channel 5 data
A:0x0B	A_LSAD_CH5_CTRL	(4:2): LSAD_CTRL_RANGE N/A	(4:2): LSAD_CTRL_RANGE (1:0): LSAD_CTRL_LSB	0x0 N/A	LSAD5 input dynamic range settings LSBs of LSAD5
A:0x0C	A_LSAD_VBAT_DATA	N/A	(7:0): A_LSAD_VBAT_DATA	N/A	Supply voltage (VBAT) data
A:0x0D	A_LSAD_VBAT_CTRL	(4:2): LSAD_CTRL_RANGE N/A	(4:2): LSAD_CTRL_RANGE (1:0): LSAD_CTRL_LSB	0x0 N/A	LSAD VBAT input dynamic range settings LSBs of LSAD_VBAT
A:0x10	A_CCR_DATA	(7:0): A_CCR_DATA	(7:0): A_CCR_DATA	0x80	Clock calibration value
A:0x11	A_CLK_CTRL	(3:0): CLK_CTRL_SYS_CLK	(3:0): CLK_CTRL_SYS_CLK	0x1	System clock frequency
A:0x12	A_INFILT_CTRL	(5:4): INFILT_CTRL_DC_REMOVE 2: INFILT_CTRL_LPF_ENABLE 1: INFILT_CTRL_LINEOUT_EBL 0: INFILT_CTRL_LINEOUT_INV	(5:4): INFILT_CTRL_DC_REMOVE 2: INFILT_CTRL_LPF_ENABLE 1: INFILT_CTRL_LINEOUT_EBL 0: INFILT_CTRL_LINEOUT_INV	0x2 0x1 0x0 0x0	DC removal filter cut-off frequency Enable/disable low pass input filter AI1/Lout pad control Lout inversion control
A:0x13	A_DIV_CLK_CTRL	(7:4): DIV_CLK_CTRL_LSAD_FREQ_CTRL (3:0): DIV_CLK_CTRL_CHARGE_PUMP_PRESCALE	(7:4): DIV_CLK_CTRL_LSAD_FREQ_CTRL (3:0): DIV_CLK_CTRL_CHARGE_PUMP_PRESCALE	0x4 0x9	Low-speed A/D sampling rate prescaler Charge pump refresh frequency
A:0x14	A_SUP_THRSH_DATA	(7:0): A_SUP_THRSH_DATA	(7:0): A_SUP_THRSH_DATA	0x7F	Battery monitor detection threshold
A:0x15	A_SUP_COUNT_DATA	N/A	(7:0): A_SUP_COUNT_DATA	N/A	Battery monitor counter value
A:0x16	A_INPUT_CTRL	(6:5): INPUT_CTRL_AD1_IN_SEL 4: INPUT_CTRL_AD1_ENABLE (2:1): INPUT_CTRL_AD0_IN_SEL 0: INPUT_CTRL_AD0_ENABLE	(6:5): INPUT_CTRL_AD1_IN_SEL 4: INPUT_CTRL_AD1_ENABLE (2:1): INPUT_CTRL_AD0_IN_SEL 0: INPUT_CTRL_AD0_ENABLE	0x2 0x0 0x0 0x1	Input for ADC1 Enable/disable ADC1 Input for ADC0 Enable/disable ADC0

TABLE 18: ANALOG CONTROL REGISTERS (CONTINUED)

Address	Register Name	Register Write	Register Read	Default	Description
A:0x17	A_IN_GAIN_CTRL	(6:4): IN_GAIN_PROG_GAIN1 (2:0): IN_GAIN_PROG_GAIN0	(6:4): IN_GAIN_PROG_GAIN1 (2:0): IN_GAIN_PROG_GAIN0	0x3 0x3	Gain value for preamplifier 1 Gain value for preamplifier 0
A:0x18	A_OUT_ATTEN_CTRL	(7:5): OUT_ATTEN_CTRL_OUT1_ATTEN 4: OUT_ATTEN_CTRL_MUTE1 (3:1): OUT_ATTEN_CTRL_OUT0_ATTEN 0: OUT_ATTEN_CTRL_MUTE0	(7:5): OUT_ATTEN_CTRL_OUT1_ATTEN 4: OUT_ATTEN_CTRL_MUTE1 (3:1): OUT_ATTEN_CTRL_OUT0_ATTEN 0: OUT_ATTEN_CTRL_MUTE0	0x0 0x1 0x0 0x1	Select attenuation value for attenuator 1 Mute/pass signal to analog output 1 Select attenuation value for attenuator 0 Mute/pass signal to analog output 0
A:0x19	A_DAC_CTRL	(7:6): DAC_CTRL_OSR (5:4): DAC_CTRL_CURRENT1 (1:0): DAC_CTRL_CURRENT0	(7:6): DAC_CTRL_OSR (5:4): DAC_CTRL_CURRENT1 (1:0): DAC_CTRL_CURRENT0	0x0 0x1 0x1	Direct digital output oversampling frequency Select current for D/A converter 1 Select current for D/A converter 0
A:0x1A	A_PSU_CTRL	4: PSU_CTRL_VSPI_MODE (1:0): PSU_CTRL_VDDC_MODE	4: PSU_CTRL_VSPI_MODE (1:0): PSU_CTRL_VDDC_MODE	0x0 0x0	SPI voltage mode Power supply sub-mode
A:0x1C	A_ADC_CUR_CTRL	7: ADC_CUR_CTRL_HIGH_FREQ_EBL (5:3): ADC_CUR_CTRL_ADC1_CURRENT (2:0): ADC_CUR_CTRL_ADC0_CURRENT	7: ADC_CUR_CTRL_HIGH_FREQ_EBL (5:3): ADC_CUR_CTRL_ADC1_CURRENT (2:0): ADC_CUR_CTRL_ADC0_CURRENT	0x0 0x3 0x3	Setting depends on MCLK frequency Current for A/D converter 1 Current for A/D converter 0
A:0x1D	A_ADC_CTRL	(5:0): ADC_CTRL_SAMPLE_FREQ	(5:0): ADC_CTRL_SAMPLE_FREQ	0x02	Sampling frequency
A:0x1E	A_OUTPUT_CTRL	7: OUTPUT_CTRL_HPWR_MODE 6: OUTPUT_CTRL_LPF_MODE_OUT 3: OUTPUT_CTRL_DAI_ENABLE 2: OUTPUT_CTRL_DAO_ENABLE 1: OUTPUT_CTRL_OD1_ENABLE 0: OUTPUT_CTRL_OD0_ENABLE	7: OUTPUT_CTRL_HPWR_MODE 6: OUTPUT_CTRL_LPF_MODE_OUT 3: OUTPUT_CTRL_DAI_ENABLE 2: OUTPUT_CTRL_DAO_ENABLE 1: OUTPUT_CTRL_OD1_ENABLE 0: OUTPUT_CTRL_OD0_ENABLE	0x0 0x0 0x0 0x0 0x0 0x0	Enable/disable high power output mode Low-pass filter cut-off Enable/disable D/A converter 1 Enable/disable D/A converter 0 Enable/disable direct digital output 1 Enable/disable direct digital output 0
A:0x1F	A_ADC_GFO_CTRL	(7:0): A_ADC_GFO_CTRL	(7:0): A_ADC_GFO_CTRL	0x65	Decimation filter gain value for channel 0
A:0x20	A_ADC_GF1_CTRL	(7:0): A_ADC_GF1_CTRL	(7:0): A_ADC_GF1_CTRL	0x65	Decimation filter gain value for channel 1
A:0x21	A_DEL_INT_CTRL	7: DEL_INT_CTRL_LEADER (2:0): DEL_INT_CTRL_DEL_INT	7: DEL_INT_CTRL_LEADER (2:0): DEL_INT_CTRL_DEL_INT	0x0 0x0	Select whether AD1 signal leads AD0 Integer sample delay
A:0x22	A_DEL_FRAC_CTRL	(5:0): DEL_FRAC_CTRL_DEL_FRAC	(5:0): DEL_FRAC_CTRL_DEL_FRAC	0x00	Fractional sample delay

TABLE 19: DIGITAL CONTROL REGISTERS

Address	Register Name	Register Write	Register Read	Default	Description
Y:0x4000	D_SPI_DATA	(15:0): D_SPI_DATA	(15:0): D_SPI_DATA	0x0000	SPI transmit/receive register
Y:0x4001	D_SPI_CTRL	7: SPI_CTRL_START 6: SPI_CTRL_RW 5: SPI_CTRL_CLKPOL 4: SPI_CTRL_CS_N (3:0): SPI_CTRL_N_BITS	7: SPI_CTRL_BUSY 6: SPI_CTRL_RW 5: SPI_CTRL_CLKPOL 4: SPI_CTRL_CS_N (3:0): SPI_CTRL_N_BITS	0x0 0x0 0x0 0x0 0x0	Start transmission/return transmission state Control read/write data from SPI Set clock polarity Level of chip select Number of bits to send, less one
Y:0x4002	D_PCM_DATA0	(15:0): D_PCM_DATA0	(15:0): D_PCM_DATA0	0x0000	PCM subframe zero
Y:0x4003	D_PCM_DATA1	(15:0): D_PCM_DATA1	(15:0): D_PCM_DATA1	0x0000	PCM subframe one
Y:0x4004	D_SSP_CFG_CTRL	15: SSP_CFG_CTRL_START (11:8): SSP_CFG_CTRL_OP (7:0): SSP_CFG_CTRL_DATA	15: SSP_CFG_CTRL_BUSY (11:8): SSP_CFG_CTRL_OP (7:0): SSP_CFG_CTRL_DATA	0x0 0x0 0x00	Set bit to start transfer/return transfer state Specify operation direction and behaviour 8-bit value to write / has been read
Y:0x4005	D_CFG_REG_CTRL	0: CFG_REG_LOCK	0: CFG_REG_LOCK	0x0	Lock control register access
Y:0x4006	D_GPIO_DATA	(15:0): D_GPIO_DATA	(15:0): D_GPIO_DATA	N/A	GPIO data
Y:0x4007	D_FATC_DATA	N/A	(15:0): D_FATC_DATA	0x1800	Pointer to oldest sample in input FIFO input processing domain
Y:0x4008	D_UART_DATA	(7:0): UART_DATA	(7:0): UART_DATA	0x00	UART data transmit / receive register
Y:0x4009	D_BLOCK_EXP_DATA	(3:0): BLOCK_EXP	(3:0): BLOCK_EXP	N/A	Block exponent used by the WOLA
Y:0x400A	D_GAIN_EXP_DATA	(3:0): GAIN_EXP	(3:0): GAIN_EXP	0x0	Gain exponent
Y:0x400B	D_MAGIC_READ	N/A	(1:0): MAGIC_READ_N_WOLA	N/A	Pending WOLA shifts
Y:0x400C	D_TWSS_DATA	(7:0): TWSS_DATA	(7:0): TWSS_DATA	0x00	TWSS transmit / receive register
Y:0x400D	D_TWSS_CTRL	N/A N/A N/A N/A 0: TWSS_CTRL_CLK_REL	4: TWSS_CTRL_STOP_RCV 3: TWSS_CTRL_ADDR_RCV 2: TWSS_CTRL_BUSY 1: TWSS_CTRL_GEN_CALL 0: TWSS_CTRL_RW	N/A N/A N/A N/A 0x0	Data/stop condition receive indicator Address/data receive indicator Interface busy indicator General call address/specific address indicator Read/write indicator/manual clock release
Y:0x400E	D_REMOTE_DATA	N/A N/A	8: REMOTE_DATA_INIT_BYTE_RCV (7:0): REMOTE_DATA	N/A N/A	First byte received indicator Remote control receive register
Y:0x400F	D_INT_CTRL	(15:12): INT_CTRL_PPIO 2: INT_CTRL_PPIO_EBL 1: INT_CTRL_AUTO_ROT 0: INT_CTRL_AUTO_ACK	(15:12): INT_CTRL_PPIO 2: INT_CTRL_PPIO_EBL 1: INT_CTRL_AUTO_ROT 0: INT_CTRL_AUTO_ACK	0x0 0x0 0x0 0x0	Interrupt priority setting Enable/disable interrupt priorities Enable/disable interrupt priority auto-rotation Enable/disable interrupt auto-acknowledge
Y:0x4010	D_DBG_AUDIO_MODE_CTRL	(1:0): DBG_AUDIO_MODE_CTRL	(1:0): DBG_AUDIO_MODE_CTRL	0x0	Control debug port audio streaming
Y:0x403F	D_PCM_CTRL	12: PCM_TX_UNDERFLOW_RESET (11:8): PCM_TX_PTR_STATE 4: PCM_RX_OVERFLOW_RESET (3:0): PCM_RX_PTR_STATE	12: PCM_TX_UNDERFLOW (11:8): PCM_TX_PTR_STATE 4: PCM_RX_OVERFLOW (3:0): PCM_RX_PTR_STATE	0x0 0xF 0x0 0xE	Transmit underflow flag/reset Transmit FIFO pointer Receive overflow flag/reset Receive FIFO pointer
Y:0x4040	D_PCM_DATA0_INDEX0	(15:0): D_PCM_DATA0_INDEX0	(15:0): D_PCM_DATA0_INDEX0	N/A	Direct PCM buffer access
Y:0x4041	D_PCM_DATA1_INDEX0	(15:0): D_PCM_DATA1_INDEX0	(15:0): D_PCM_DATA1_INDEX0	N/A	Direct PCM buffer access
Y:0x4042	D_PCM_DATA0_INDEX1	(15:0): D_PCM_DATA0_INDEX1	(15:0): D_PCM_DATA0_INDEX1	N/A	Direct PCM buffer access

TABLE 19: DIGITAL CONTROL REGISTERS (CONTINUED)

Address	Register Name	Register Write	Register Read	Default	Description
Y:0x4043	D_PCM_DATA1_INDEX1	(15:0): D_PCM_DATA1_INDEX1	(15:0): D_PCM_DATA1_INDEX1	N/A	Direct PCM buffer access
Y:0x4044	D_PCM_DATA0_INDEX2	(15:0): D_PCM_DATA0_INDEX2	(15:0): D_PCM_DATA0_INDEX2	N/A	Direct PCM buffer access
Y:0x4045	D_PCM_DATA1_INDEX2	(15:0): D_PCM_DATA1_INDEX2	(15:0): D_PCM_DATA1_INDEX2	N/A	Direct PCM buffer access
Y:0x4046	D_PCM_DATA0_INDEX3	(15:0): D_PCM_DATA0_INDEX3	(15:0): D_PCM_DATA0_INDEX3	N/A	Direct PCM buffer access
Y:0x4047	D_PCM_DATA1_INDEX3	(15:0): D_PCM_DATA1_INDEX3	(15:0): D_PCM_DATA1_INDEX3	N/A	Direct PCM buffer access
Y:0x4048	D_PCM_DATA0_INDEX4	(15:0): D_PCM_DATA0_INDEX4	(15:0): D_PCM_DATA0_INDEX4	N/A	Direct PCM buffer access
Y:0x4049	D_PCM_DATA1_INDEX4	(15:0): D_PCM_DATA1_INDEX4	(15:0): D_PCM_DATA1_INDEX4	N/A	Direct PCM buffer access
Y:0x404A	D_PCM_DATA0_INDEX5	(15:0): D_PCM_DATA0_INDEX5	(15:0): D_PCM_DATA0_INDEX5	N/A	Direct PCM buffer access
Y:0x404B	D_PCM_DATA1_INDEX5	(15:0): D_PCM_DATA1_INDEX5	(15:0): D_PCM_DATA1_INDEX5	N/A	Direct PCM buffer access
Y:0x404C	D_PCM_DATA0_INDEX6	(15:0): D_PCM_DATA0_INDEX6	(15:0): D_PCM_DATA0_INDEX6	N/A	Direct PCM buffer access
Y:0x404D	D_PCM_DATA1_INDEX6	(15:0): D_PCM_DATA1_INDEX6	(15:0): D_PCM_DATA1_INDEX6	N/A	Direct PCM buffer access
Y:0x404E	D_PCM_DATA0_INDEX7	(15:0): D_PCM_DATA0_INDEX7	(15:0): D_PCM_DATA0_INDEX7	N/A	Direct PCM buffer access
Y:0x404F	D_PCM_DATA1_INDEX7	(15:0): D_PCM_DATA1_INDEX7	(15:0): D_PCM_DATA1_INDEX7	N/A	Direct PCM buffer access

TABLE 20: CONFIGURATION REGISTERS

Address	Register Name	Register Write	Register Read	Default	Description
Y:0x8000	D_IO_PROC_CFG	13: IO_PROC_CFG_AUTOMUTE 12: IO_PROC_CFG_ENABLE (7:6): IO_PROC_CFG_WIN (5:3): IO_PROC_CFG_BLK 2: IO_PROC_CFG_CHAN_SEP 1: IO_PROC_CFG_OUTPUT_STEREO 0: IO_PROC_CFG_INPUT_STEREO	13: IO_PROC_CFG_AUTOMUTE 12: IO_PROC_CFG_ENABLE (7:6): IO_PROC_CFG_WIN (5:3): IO_PROC_CFG_BLK 2: IO_PROC_CFG_CHAN_SEP 1: IO_PROC_CFG_OUTPUT_STEREO 0: IO_PROC_CFG_INPUT_STEREO	0x0 0x0 0x0 0x0 0x0 0x0 0x0	Enable/disable auto-mute feature Enable/disable IOP Analysis window length Input block size FIFO data organization Output FIFO mode Input FIFO mode
Y:0x8001	D_SPI_CFG	3: SPI_CFG_ENABLE (2:0): SPI_CFG_PRESCALE	3: SPI_CFG_ENABLE (2:0): SPI_CFG_PRESCALE	0x0 0x6	Enable/disable SPI port Select prescale factor used to generate SPI clock
Y:0x8002	D_PCM_CFG	(10:8): PCM_RX_INT_CFG 7: PCM_TX_INT 6: PCM_RX_INT 5: PCM_FRAME_WIDTH 4: PCM_FRAME_ALIGN 3: PCM_BIT_ORDER 2: PCM_SUBFRAME 1: PCM_ENABLE 0: PCM_SLAVE	(10:8): PCM_RX_INT_CFG 7: PCM_TX_INT 6: PCM_RX_INT 5: PCM_FRAME_WIDTH 4: PCM_FRAME_ALIGN 3: PCM_BIT_ORDER 2: PCM_SUBFRAME 1: PCM_ENABLE 0: PCM_SLAVE	0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0	Samples between receive interrupts Transmit interrupt enable Receive interrupt enable Width of frame signal Alignment of frame signal Transmission of MSB first or LSB first Frame subframes or frames Enable/disable PCM interface Master/slave mode configuration
Y:0x8003	D_UART_CFG	1: UART_CFG_ENABLE 0: UART_CFG_PRESCALE	1: UART_CFG_ENABLE 0: UART_CFG_PRESCALE	0x0 0x0	Enable/disable UART Enable/disable prescaler
Y:0x8004	D_UART_SPEED_CFG	(15:0): D_UART_SPEED_CFG	(15:0): D_UART_SPEED_CFG	0x0000	UART speed

TABLE 20: CONFIGURATION REGISTERS (CONTINUED)

Address	Register Name	Register Write	Register Read	Default	Description
Y:0x8005	D_TIMER_CFG	15: TIMER_CFG_MODE (14:12): TIMER_CFG_PRESCALE (11:0): TIMER_CFG_DELAY	15: TIMER_CFG_MODE (14:12): TIMER_CFG_PRESCALE (11:0): TIMER_CFG_DELAY	0x0 0x0 0x000	One-shot or free-run System clock cycles per timer clock cycle Timer expiration setting
Y:0x8006	D_GPIO_DIR_CFG	(15:0): D_GPIO_DIR_CFG	(15:0): D_GPIO_DIR_CFG	0xFFFF	Direction of GPIO (or multiplexed functionality) pin
Y:0x8007	D_GPIO_PIN_CFG	15: GPIO_PIN_CFG_I2S 14: GPIO_PIN_CFG_REMOTE 12: GPIO_PIN_CFG_UART 11: GPIO_PIN_CFG_PCM 10: GPIO_PIN_CFG_UCLK 9: GPIO_PIN_CFG_LSAD3 8: GPIO_PIN_CFG_LSAD4 7: GPIO_PIN_CFG_LSAD5 6: GPIO_PIN_CFG_LSAD2 5: GPIO_PIN_CFG_LSAD1 4: GPIO_PIN_CFG_LSAD0 3: GPIO_PIN_CFG_CLKRST_EBL	15: GPIO_PIN_CFG_I2S 14: GPIO_PIN_CFG_REMOTE 12: GPIO_PIN_CFG_UART 11: GPIO_PIN_CFG_PCM 10: GPIO_PIN_CFG_UCLK 9: GPIO_PIN_CFG_LSAD3 8: GPIO_PIN_CFG_LSAD4 7: GPIO_PIN_CFG_LSAD5 6: GPIO_PIN_CFG_LSAD2 5: GPIO_PIN_CFG_LSAD1 4: GPIO_PIN_CFG_LSAD0 3: GPIO_PIN_CFG_CLKRST_EBL	0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0	Enable/disable use of I ² S interface Enable/disable use of wireless receiver interface Enable/disable use of UART interface Enable/disable use of PCM interface Enable/disable user clock output Enable/disable use of LSAD3 Enable/disable use of LSAD4 Enable/disable use of LSAD5 Enable/disable use of LSAD2 Enable/disable use of LSAD1 Enable/disable use of LSAD0 Enable/disable clock synchronization
Y:0x8008	D_TWSS_CFG	13: TWSS_CFG_STOP_INT_EBL 8: TWSS_CFG_AUTO_CLK_RELEASE 7: TWSS_CFG_ENABLE (6:0): TWSS_CFG_SLAVE_ID	13: TWSS_CFG_STOP_INT_EBL 8: TWSS_CFG_AUTO_CLK_RELEASE 7: TWSS_CFG_ENABLE (6:0): TWSS_CFG_SLAVE_ID	0x0 0x0 0x0 0x00	Enable/disable stop condition interrupt Auto/manual clock release Enable/disable TWSS interface Slave ID
Y:0x8009	D_CLKSEL_CFG	6: CLKSEL_CFG_STANDBY_CLK (5:4): CLKSEL_CFG_EXTCLK_SEL (1:0): CLKSEL_CFG_UCLK_SEL	6: CLKSEL_CFG_STANDBY_CLK (5:4): CLKSEL_CFG_EXTCLK_SEL (1:0): CLKSEL_CFG_UCLK_SEL	0x0 0x0 0x0	Select the source for SYS_CLK Select the behaviour of the EXTCLK I/O Pin Select the source for UCLK
Y:0x800A	D_WATCHDOG_CFG	(3:0): WATCHDOG_CFG_TIMEOUT	(3:0): WATCHDOG_CFG_TIMEOUT	0xF	Timeout for watchdog reset
Y:0x800B	D_ACCESS_CFG	2: ACCESS_CFG_COMPATIBLE_MODE 0: ACCESS_CFG_ACCESS_MODE	2: ACCESS_CFG_COMPATIBLE_MODE 0: ACCESS_CFG_ACCESS_MODE	0x0 0x1	Status byte is SK1 compatible Restrict/unrestrict access to Orela 4500
Y:0x800C	D_REMOTE_CFG	3: REMOTE_CFG_BURST_FILTER_ENABLE (2:0): REMOTE_CFG_PRESCALE	3: REMOTE_CFG_BURST_FILTER_ENABLE (2:0): REMOTE_CFG_PRESCALE	0x1 0x0	Enable/disable remote burst filter Prescale value for the remote control port
Y:0x800D	D_CHIP_VERSION	N/A	(7:0): CHIP_VERSION	N/A	Chip version ID
Y:0x800E	D_CLKDIV_CFG	(14:8): CLKDIV_CFG_PCLK_PRESCALE (6:0): CLKDIV_CFG_MCLK_PRESCALE	(14:8): CLKDIV_CFG_PCLK_PRESCALE (6:0): CLKDIV_CFG_MCLK_PRESCALE	0x00 0x00	Select prescale factor used to generate PCLK Select prescale factor used to generate MCLK
Y:0x800F	D_USRCLKDIV_CFG	(11:0): USRCLKDIV_CFG_USRCLK_PRESCALE	(11:0): USRCLKDIV_CFG_USRCLK_PRESCALE	0x000	Select prescale factor used to generate USRCLK
Y:0x8010	D_WOLADIV_CFG	(9:0): WOLADIV_CFG_WOLACLK_PRESCALE	(9:0): WOLADIV_CFG_WOLACLK_PRESCALE	0x000	Select prescale factor used to generate WOLACLK

TABLE 20: CONFIGURATION REGISTERS (CONTINUED)

Address	Register Name	Register Write	Register Read	Default	Description
Y:0x8011	D_STANDBY_CFG	1: STANDBY_CFG_CLK 0: STANDBY_CFG_ANALOG_DISABLE	1: STANDBY_CFG_CLK 0: STANDBY_CFG_ANALOG_DISABLE	0x0 0x0	Enable/disable 30kHz power management clock Disable/enable analog blocks
Y:0x8012	D_GPIO_INT_CFG	(13:10): GPIO_INT_CFG_SRC_A (9:7): GPIO_INT_CFG_EVENT_A (6:3): GPIO_INT_CFG_SRC_B (2:0): GPIO_INT_CFG_EVENT_B	(13:10): GPIO_INT_CFG_SRC_A (9:7): GPIO_INT_CFG_EVENT_A (6:3): GPIO_INT_CFG_SRC_B (2:0): GPIO_INT_CFG_EVENT_B	0x0 0x0 0x0 0x0	Select GPIO line for A Trigger event for A Select GPIO line for B Trigger event for B

TABLE 21: RCORE DIGITAL CONTROL REGISTERS

Register Name	Register write	Register read	Default	Description
EXT3	(15:0): DEBUG_PORT_DATA	(15:0): DEBUG_PORT_DATA	N/A	Debug port communication register
D_INT_STATUS	12: INT_GPIO 11: INT_EXT3_TX 10: INT_EXT3_RX 9: INT_REMOTE 8: INT_TWSS 7: INT_SPI 6: INT_WATCHDOG 5: INT_TIMER 4: INT_UART_TX 3: INT_UART_RX 2: INT_PCM 1: INT_IO_BLOCK_FULL 0: INT_WOLA_DONE	12: INT_GPIO 11: INT_EXT3_TX 10: INT_EXT3_RX 9: INT_REMOTE 8: INT_TWSS 7: INT_SPI 6: INT_WATCHDOG 5: INT_TIMER 4: INT_UART_TX 3: INT_UART_RX 2: INT_PCM 1: INT_IO_BLOCK_FULL 0: INT_WOLA_DONE	0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0	Acknowledge/status of interrupt Acknowledge/status of interrupt Acknowledge/status of interrupt Acknowledge/status of interrupt Acknowledge/status of interrupt Acknowledge/status of interrupt Acknowledge/status of interrupt Acknowledge/status of interrupt Acknowledge/status of interrupt Acknowledge/status of interrupt Acknowledge/status of interrupt Acknowledge/status of interrupt Acknowledge/status of interrupt Acknowledge/status of interrupt

TABLE 21: R-CORE DIGITAL CONTROL REGISTERS (CONTINUED)

Register Name	Register write	Register read	Default	Description
D_INT_EBL	12: INT_GPIO 11: INT_EXT3_TX 10: INT_EXT3_RX 9: INT_REMOTE 8: INT_TWSS 7: INT_SPI 6: INT_WATCHDOG 5: INT_TIMER 4: INT_UART_TX 3: INT_UART_RX 2: INT_PCM 1: INT_IO_BLOCK_FULL 0: INT_WOLA_DONE	12: INT_GPIO 11: INT_EXT3_TX 10: INT_EXT3_RX 9: INT_REMOTE 8: INT_TWSS 7: INT_SPI 6: INT_WATCHDOG 5: INT_TIMER 4: INT_UART_TX 3: INT_UART_RX 2: INT_PCM 1: INT_IO_BLOCK_FULL 0: INT_WOLA_DONE	0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0	Enable/disable interrupt Enable/disable interrupt Enable/disable interrupt Enable/disable interrupt Enable/disable interrupt Enable/disable interrupt Enable/disable interrupt Enable/disable interrupt Enable/disable interrupt Enable/disable interrupt Enable/disable interrupt Enable/disable interrupt
D_SYS_CTRL	8: N/A 7: N/A 6: SYS_CTRL_IOP_RESET 5: SYS_CTRL_WOLA_RESET 4: SYS_CTRL_WATCHDOG_RESET 3: SYS_CTRL_TIMER_START 2: SYS_CTRL_WOLA_START (1:0) : SYS_CTRL_WOLA_FUNCTION	8: SYS_CTRL_BLK_PROCESSED 7: SYS_CTRL_WOLA_BUSY 6: N/A 5: N/A 4: N/A 3: SYS_CTRL_TIMER_BUSY 2: N/A (1:0) : SYS_CTRL_WOLA_FUNCTION	0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0	IOP auto-mute flag WOLA busy indicator IOP reset WOLA reset Watchdog acknowledge Start timer / timer busy indicator Start WOLA filterbank function WOLA filterbank function

This page left blank for printing purposes.

Corporate Head Office

Dspfactory Ltd.
611 Kumpf Drive, Unit 200
Waterloo, Ontario
Canada N2V 1K8
Tel: +1 519 884 9696
Fax: +1 519 884 0228

European Office

Dspfactory SA
Champs-Montants 12a
2074 Marin
Switzerland
Tel: +41 32 755 74 30
Fax: +41 32 755 74 01

Website

www.dspfactory.com

E-mail

info@dspfactory.com

Copyright © 2004 Dspfactory Ltd. All rights reserved.
Dspfactory, SignaKlara and Orela are either
trademarks or registered trademarks of Dspfactory
Ltd. All other brands, product names and company
names mentioned herein may be trademarks or
registered trademarks of their respective holders.
Specifications subject to change. Printed in Canada.